INTERPRETING A MUSICAL SCORE SHEET

Sheenah A. Ancheta

TABLE OF CONTENTS

01

INTRODUCTION

Interpreting a musical score sheet

02

ALGORITHM

Process implemented in Scilab, with code snippets

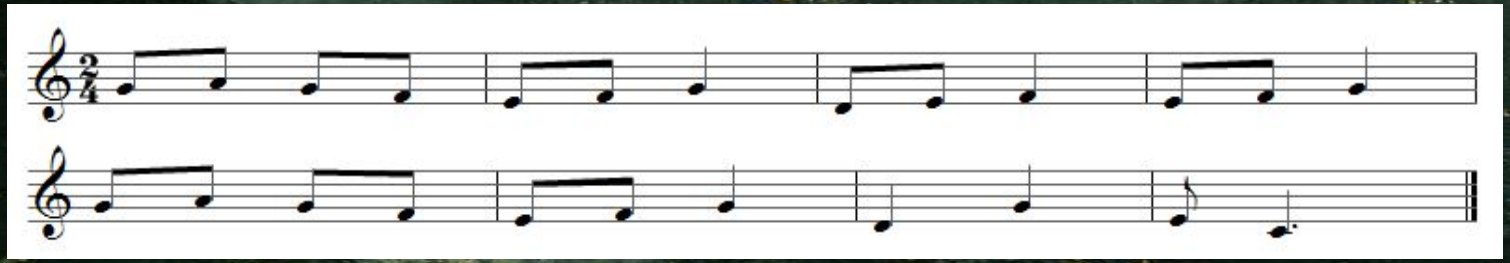
03

RESULTS

SELF-EVALUATION

INTRODUCTION

MUSICAL SCORE SHEET



LONDON BRIDGE MUSICAL SCORE SHEET

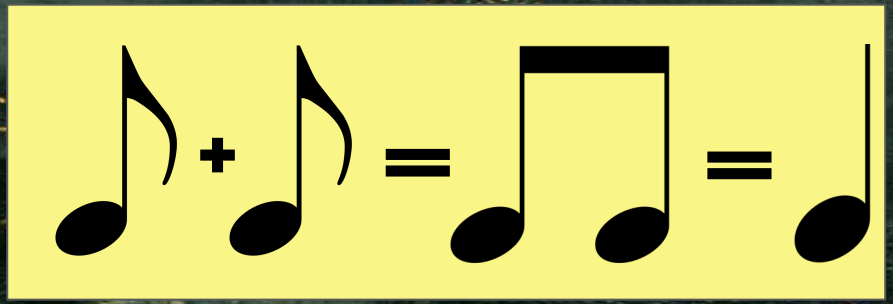
SYMBOLS OF NOTATION

symbols seen in the London Bridge musical score sheet

TREBLE CLEF | STAFF | TIME SIGNATURE | QUARTER NOTE | EIGHTH NOTE

[1] https://bit.ly/2pdS2cR

NOTE VALUES



It should be noted that an eighth note is just a quarter notes with a **flag** (left). Two eighth notes are equal to two quarter notes combined by a **beam** (middle). We signify faster notes with either *flags* or *beams* between the notes. A flag halves the value of a note, so a single flag signifies ½ of a quarter note. Therefore, an eighth note is the half of a quarter note. Similarly, a beam also halves the value of the combination of notes it is beaming. However, unlike flags, beams keep the notation in a musical score sheet less cluttered.[1] In conclusion, 2 eighth notes and two quarter notes combined by a beam is equal to a single quarter note.

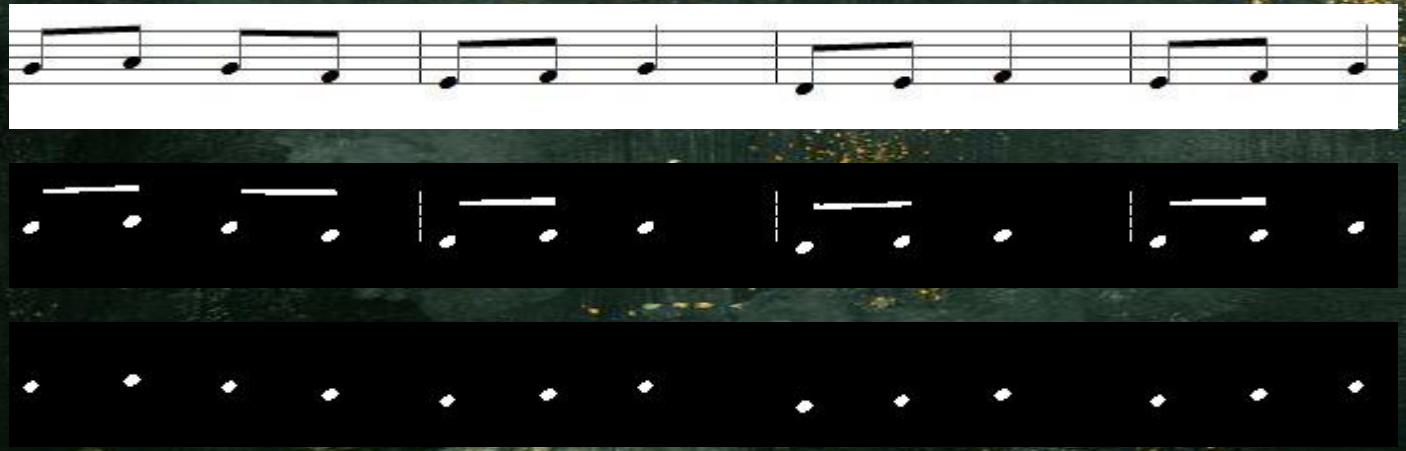
[1] https://bit.ly/2pdS2cR

ALGORITHM

The code snippets shown here are constructed in Scilab with a guide from Mr. Harold Co's blog[2].

[2] https://bit.ly/2nUaLtX

OBTAINING THE BLOBS



The top figure shows a cropped version of the musical score sheet of London Bridge in which only the first row was included. The image in the middle shows the result when a threshold of 0.1 was applied. It can be observed that although the vertical lines from the notes (stem) were eliminated, the beams and the measure lines still remained. To fix this, *OpenImage( )* was used and the resulting image (bottom figure) of just pure blobs was obtained.

```
7  //Morphological Operation
8  se=CreateStructureElement('circle', 2)
9  notes = OpenImage(notes,se)
10 imshow(notes)
```

CENTROID OF A BLOB

This image only shows us the blobs obtained after thresholding and morphological operation. To be able to play this music, we must assign the pitch for each note and apply the time signature given to achieve the right song timing. We do this by obtaining the x- and y- distances *(to be further discussed on slides 11 and 13)*.

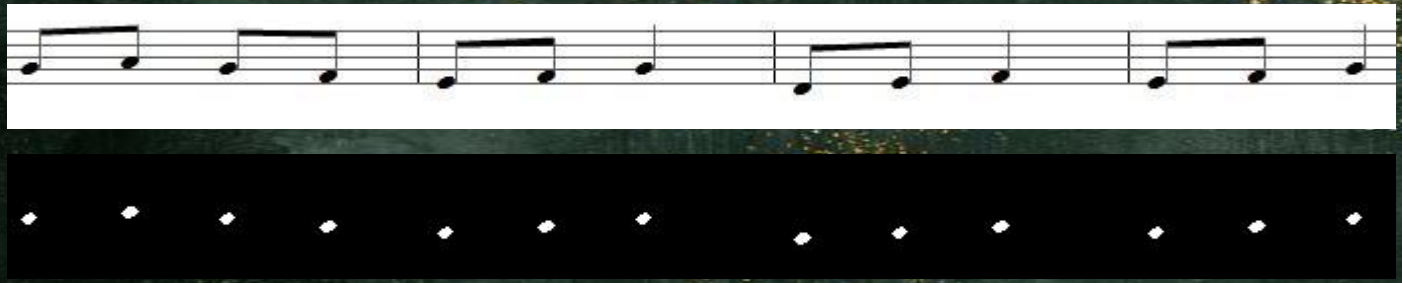
To be more accurate in handling the distances, the best way to do is to obtain the centroids of each blob. For example, we obtain the distance between two notes using their centroids as start and end points.

BLOB CENTROID

CODE SNIPPET

```
//Centroid of each blob
Object = SearchBlobs(notes);
x_cent = zeros(1,max(Object))
y_cent = zeros(1,max(Object))

for i = 1: max(Object)
    [y,x] = find(Object==i)
    xmean = mean(x)
    ymean = mean(y)
    x_cent(i) = xmean
    y_cent(i) = ymean
end
```



Since we have two different notes, quarter note and eighth note, there has to be a way for us to distinguish the two when we only see the blobs:

Distance between two notes!

The x-pixel distance between two eighth notes and an eighth note to a quarter note is 50px while from a quarter note to an eighth note, the distance is 78 px. So now we know that if we see a smaller distance, it is either between two eighth notes or between a quarter note and a half note. A bigger distance signify a half note then a quarter note.

Note on the x-pixel distance between two notes:

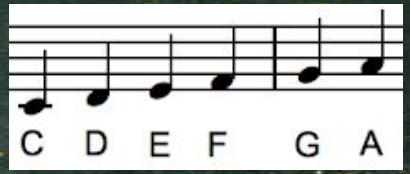
- 2 eighth notes: 50 px
- eighth note to quarter note: 50 px
- quarter note to eighth note: 78 px

CODE SNIPPET

```
50 //TIMING
51 spacing=diff(x_cent)
52 timing=zeros(1,size(x_cent,2))
53 for j=1:size(spacing,2)
54     if spacing(j)>60
55         timing(j)=2 //quarter
56     end
57     if spacing(j)<60
58         timing(j)=1 //eighth
59     end
60 end
61 timing(1,13)= 2 //quarter
```

I set my spacing condition to < or > 60 wherein a spacing of >60 between two notes means that the second note is a quarter note while a spacing of <60 signifies an eighth note. A timing value of 2 represents a quarter note while a shorter timing value of 1 represents an eighth note.

At line 61, I just set the 13th or the last note to a quarter note, since there is no note after it.



Now that we know the timing, we then need to assign a particular sound/pitch for each note. We do this by obtaining the:

Position along the staff!

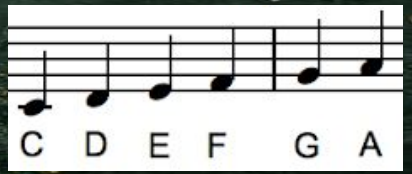
A lower-pitched note has a lower position along the staff while a higher-pitched note has a higher position. In terms of y-pixel distance, a lower pitched note has a higher y-pixel value. Similarly, a higher pitched note has a lower y-pixel value.

CODE SNIPPET

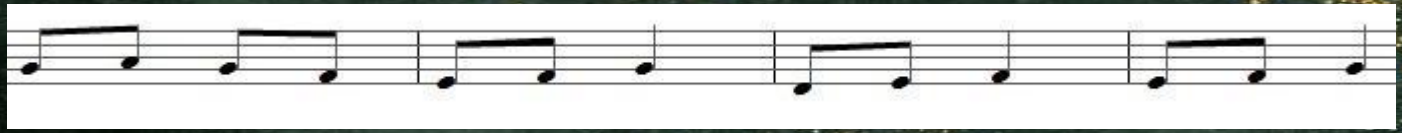
```
//NOTE PITCH
note=zeros(1,size(y_cent,2))
for j=1:size(y_cent,2)
    if y_cent(1,j)>44 & y_cent(1,j)<46
        note(1,j) = 261.63/2 //C
    end
    if y_cent(1,j)>41 & y_cent(1,j)<43
        note(1,j) = 293.66/2 //D
    end
    if y_cent(1,j)>38 & y_cent(1,j)<40
        note(1,j) = 329.63/2 //E
    end
    if y_cent(1,j)>35 & y_cent(1,j)<37
        note(1,j) = 349.23/2 //F
    end
    if y_cent(1,j)>32 & y_cent(1,j)<34
        note(1,j) = 392/2 //G
    end
    if y_cent(1,j)>29 & y_cent(1,j)<31
        note(1,j) = 440/2 //A
    end
end
```

It should be noted that here, I took note of the possible deviations in the allocation of centroids along the y-direction. As an example, for note C, I found out through paint that the blob centroid along the y-direction is 45 px. To account for errors, I made the range of having ±1 from the original value of 45px -- thus, my C note ranges from 44 px to 46 px along the staff.

I divided all the pitch values by 2 to play the song in lower frequency, and thus, lower-pitched sounds.



PLAYING THE MUSIC



Since the pitch for each note was already obtained and the time signature was already applied to achieve the right timing for the selected the song, we could now move forward to the last step which is to play the music.

Two ways are used to play the music-- with and without an envelope. An envelope is used for synthesizers to obtain a more natural sound. We use this for a more beautiful output since it stimulates the volume envelopes of a real instrument.[3]

```
64
65 //PLAYING THE MUSIC
1  function n = note_function(f, t)
2      n = sin(2*%pi*f*linspace(0,t,8192*t));
3  endfunction;
69
```

function used to play music without envelope

```
70
1  function n = note_function(f, t)
2      n = sin(2*%pi*f*linspace(0,t,8192*t));
3      line1 = linspace(0,1,410*t);
4      line2 = linspace(1,1,819*t);
5      line3 = linspace(1,0.9,819*t);
6      line4 = linspace(0.9,0.45,5734*t);
7      line5 = linspace(0.45,0,410*t);
8      envelope = [line1,line2,line3,line4,line5];
9      n = n.*envelope
10 endfunction;
81
```

function used to play music with envelope

[3] https://bit.ly/2n5z8nQ

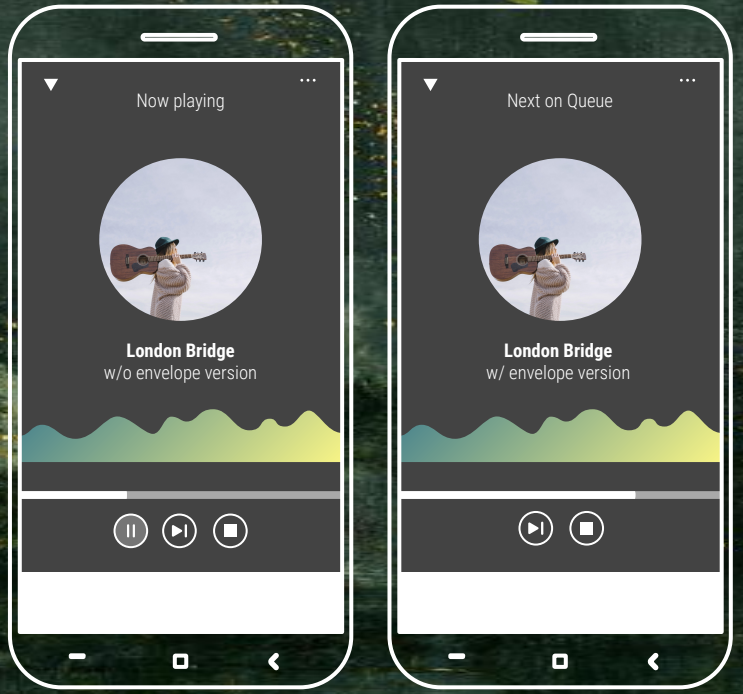
RESULTS

FINAL MUSIC

WITHOUT ENVELOPE

song link:

https://bit.ly/2nQqo5z



WITH ENVELOPE

song link:

https://bit.ly/2oJQU0A

SELF EVALUATION

5	5	1
TECHNICAL CORRECTNESS	QUALITY OF PRESENTATION	INITIATIVE