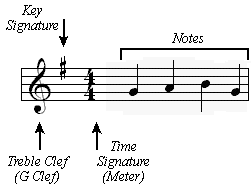
Music Score Reader

*Jacinta Samir 206562*

A music score is used denote pieces and read by musicians so that they may play them correctly. When a musician practices, only reading the notes may not suffice, and it is helpful for them to be able to hear the piece of music. Music playback can also assist composers, as they must materialize how their compositions sound like.

There are many elements that comprise a music score, such as the horizontal lines, known as ledger lines, the staff (each grouping of 5 lines), and symbols that define the pitches of the notes and how they are played (the clef, the key signature, the time signature, accidentals, and many more).



The location of each note head relative to the staff lines determines not only its alphabetical notation, but also which octave it belongs to. For the scope of this project, the jingle bells sheet music was worked on, which depicts only a treble clef, common time (4/4 time signature), and is in the scale of C major. The range of notes included was from C4 to G5, which is a larger range than that of the jingle bells score. The aforementioned properties also apply to the twinkle twinkle little star sheet music, but regrettably this was not performed on. The range of notes is as such:



The note lengths or durations included in the test cases are quarter notes, half notes, and whole notes, and other lengths were not looked at. These are, from left to right: whole, half, and quarter notes.



The input image was prepped by binarization, and then inversion, to make the notes and staff the points of interest. The image was then also cropped to remove unwanted areas.

jb = imread("JingleBells.bmp");

jbg = rgb2gray(jb);

jbb = imbinarize(jbg);

jbinv = ~ jbb;

nexttile;

imshow(jbinv);

[height, width] = size(jbinv);

jbinv = imcrop(jbinv, [95,100,width-95,height-150]);

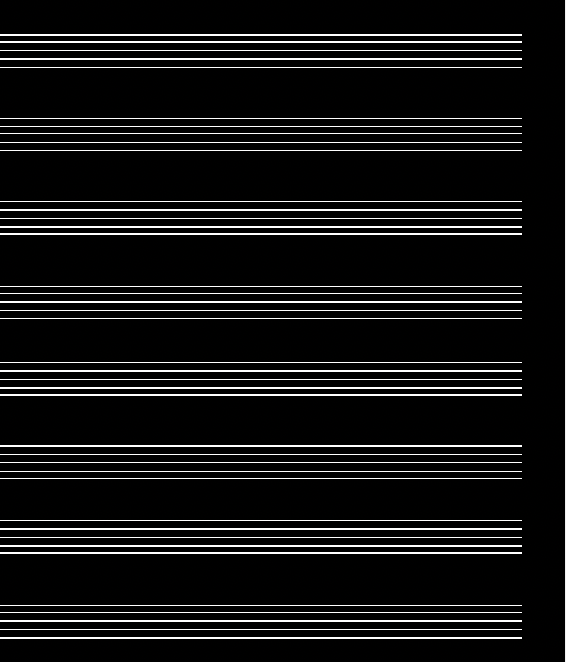


The beginning step was to remove the staff lines, in order to be able to later process the notes themselves. This was done by opening (erosion followed by dilation) the image using a long line-shaped structuring element, with degree 0.

se = strel("line", 20, 0);

jboh = imopen(jbinv, se);

jbs = jbinv - jboh;

The vertical bar lines were also removed in the same manner, except with a 90 degree angle, and a longer length, as to not remove any note stems.

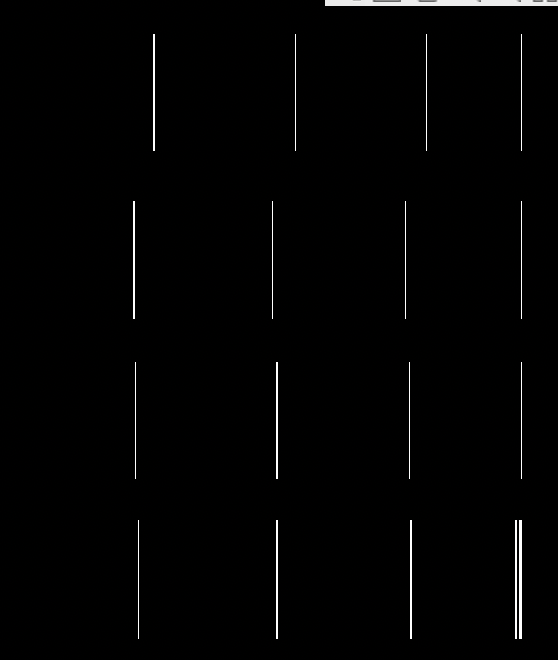
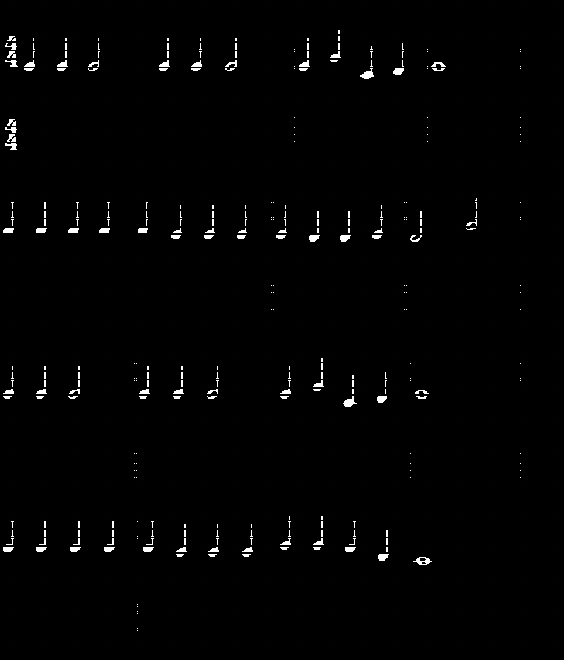
se = strel("line", 40, 90);

jbov = imopen(jbinv, se);

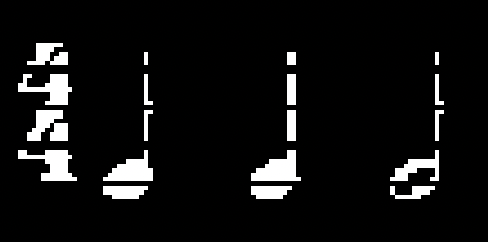
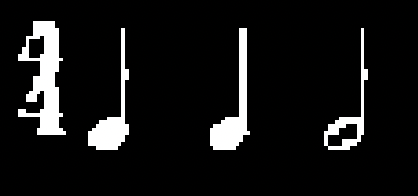
nexttile;

imshow(jbov);

jbss = jbs - jbov;

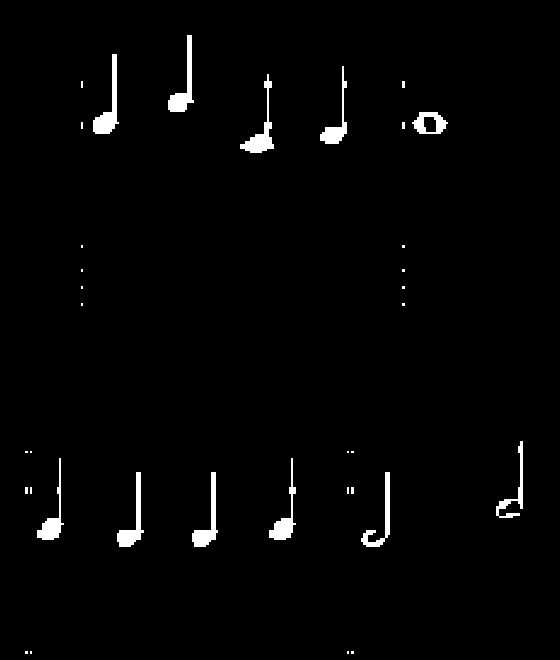
 

Removing the staff lines has created cuts along the notes, so another line structuring element (of short length) will be used to repair these, this time by closing the image (dilation followed by erosion).

In the areas in which the staff lines (horizontal) met the bar lines (vertical), white spots can be seen. To remove these occurrences of intersection, bwareaopen was used to get rid of any small pixel areas.

jbco = bwareaopen(jbc,4);

Using the connected components of the binary image, bounding boxes were plotted atop all notes and items.

cc = bwconncomp(jbco);

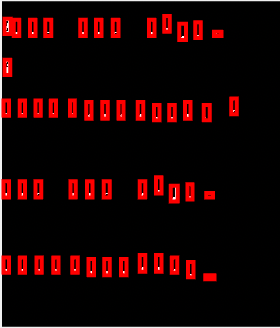
sb = regionprops(cc,"BoundingBox");

for i = 1 : length(sb)

bb = sb(i).BoundingBox;

rectangle('Position', bb, 'EdgeColor', 'r', 'LineWidth', 2);

end

Now that the notes are located, they must be identified. The first requirement in classifying a note is determining its length (whether it be a quarter, half or whole note, or perhaps not a note at all, such as the 4/4 time signature seen at the start). Identification of these notes was first attempted using a fast Fourier transform, however the results were unsuccessful in classifying the note lengths. The code for this can still be found, commented out of the m file. Thus, a different approach was used.

Firstly, the bottom ledger line of each treble clef staff was located by iterating on the staff lines found earlier. These lines were then used as a guide line in sectioning the image line-by-line. The locations of the bottom lines were used to crop the image into several sub-images. The bass clef staff was ignored, as it is empty.

staffcc = bwconncomp(jboh);

staffbb = regionprops(staffcc, "BoundingBox");

count = 0;

for i = 5 : 10 : length(staffbb)

count = count + 1;

bb = staffbb(i).BoundingBox;

rectangle('Position', bb, 'EdgeColor', 'r', 'LineWidth', 2);

staffs{count} = bb;

end

for i = 1 : length(staffs)

bb = staffs{i};

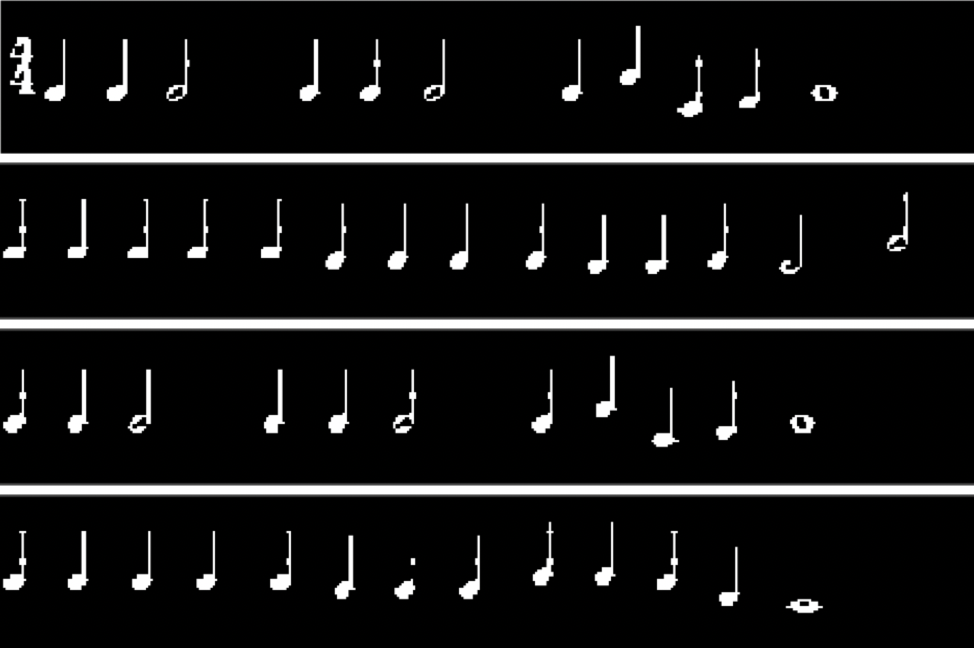
bb(2) = bb(2) - 50;

bb(4) = bb(4) + 80;

imgh{i} = imcrop(jbco, bb);

ledger{i} = imcrop(jboh, bb);

end

Each of these strips of images were then looped over, finding once again their components’ bounding boxes, and sorting them based on the x-coordinate, as to make sure notes are read from left to right. Each component was then checked to determine its length (quarter, half, whole, or not a note). This was done as such: the 4/4 time signature is significantly larger in area than any note, and so if the component has a large pixel area, it can be classified as not a note. The whole note has no stem, and so its bounding box’s height will be small, and thus any note with a small height is a whole note. The distinction between quarter and half notes was not as simple, as they are similar in size and area, and thus a different method was used. They were cropped to give new images, and these images were then eroded with a disk structuring element. The difference between quarter and half notes is the filling of the note head. Since the half notes are unfilled, eroding completely removed the entire note, but the filled quarter notes left a couple pixels in the center of their heads. Thus, the presence of at least one white pixel (if the maximum value of pixels is not 0) means that this note is a quarter note. This comparison was successful in classifying the notes, and these values were then stored.

for i = 1 : length(imgh)

bb = regionprops(imgh{i},"BoundingBox", "Extent", "Area");

bounds = vertcat(bb.BoundingBox);

[~, sortorder] = sortrows(bounds);

bb = bb(sortorder);

for j = 1 : length(bb)

if mod(j,2) == 0

bb2 = bb(j).BoundingBox;

rectangle('Position', bb2, 'EdgeColor', 'r', 'LineWidth', 2);

else

bb2 = bb(j).BoundingBox;

rectangle('Position', bb2, 'EdgeColor', 'g', 'LineWidth', 2);

end

if bb(j).Area > 150

noteLength{j} = "not note";

elseif bb(j).BoundingBox(4) <= 10

noteLength{j} = "whole";

else

pos = bb(j).BoundingBox;

note = imcrop(imgh{i}, [pos(1),pos(2),pos(3),pos(4)]);

circ = imerode(note, strel('disk', 3));

mxv = max(circ);

if mxv == 0

noteLength{j} = "half";

else

noteLength{j} = "quarter";

end

end

lengthStrip{i} = cat(1,noteLength);

end

allLengths = cat(1,lengthStrip);

end



The next characteristic about a note that must be identified is its pitch. This is determined by its position on the staff (i.e. whether it lies on/between/above/below which ledger lies). The first step in doing so was identifying the ledger lines (that were previously removed) for each strip image.

for i = 1 : length(ledger)

lc = regionprops(ledger{i},"BoundingBox","Centroid");

for j = 1 : length(lc)

bb = lc(j).BoundingBox;

if mod(j,2) == 0

rectangle('Position', bb, 'EdgeColor', 'r', 'LineWidth', 2);

else

rectangle('Position', bb, 'EdgeColor', 'g', 'LineWidth', 2);

end

end

end



To retrieve the position of each component relative to which lines, the strip images were looped on. For each image, the bounding boxes of the lines were located, as well as those for the components, which were sorted from left to right once more. These components were then also looped on, and for each one their center y-axis position was located (center of the note head, not the entire note, found by manipulating the coordinate of the bounding box to retrieve a point as low as the bounding box is wide). This center position can then be compared with the centroids of the ledger lines, more specifically their y-coordinate. This comparison was done for notes on each line, between two lines, above the top line, as well as multiple notes below the bottom line (via creating an imaginary sixth ledger line on the bottom, as not all notes below the staff are the same). A margin of error of 0.25 was also given, as it is unlikely that the centers of the notes would perfectly coincide with the lines. The resulting values were stored. Unfortunately, identifying the pitches of the notes was unsuccessful.

for i = 1 : length(imgh)

bbl = regionprops(ledger{i},"BoundingBox");

bb = regionprops(imgh{i},"BoundingBox");

bounds = vertcat(bb.BoundingBox);

[~, sortorder] = sortrows(bounds);

bb = bb(sortorder);

for j = 1 : length(bb)

pos = bb(j).BoundingBox;

cent = pos(2)+(pos(4)-pos(3));

centl = regionprops(ledger{i},"Centroid");

space = centl(1).Centroid(2)-centl(2).Centroid(2);

cent6 = centl(5).Centroid(2) - space;

if cent <= (centl(1).Centroid(2)+centl(1).Centroid(2)\*0.25) && cent >= (centl(1).Centroid(2)-centl(1).Centroid(2)\*0.25)

notePitch{j} = "F5";

elseif cent <= (centl(2).Centroid(2)+centl(2).Centroid(2)\*0.25) && cent >= (centl(2).Centroid(2)-centl(2).Centroid(2)\*0.25)

notePitch{j} = "D5";

elseif cent <= (centl(3).Centroid(2)+centl(3).Centroid(2)\*0.25) && cent >= (centl(3).Centroid(2)-centl(3).Centroid(2)\*0.25)

notePitch{j} = "B4";

elseif cent <= (centl(4).Centroid(2)+centl(4).Centroid(2)\*0.25) && cent >= (centl(4).Centroid(2)-centl(4).Centroid(2)\*0.25)

notePitch{j} = "G4";

elseif cent <= (centl(5).Centroid(2)+cen(5).Centroid(2)\*0.25) && cent >= (centl(5).Centroid(2)-centl(5).Centroid(2)\*0.25)

notePitch{j} = "E4";

elseif cent <= (cent6+cent6\*0.25) && cent >= (cent6+cent6\*0.25)

notePitch{j} = "C4";

elseif cent > centl(1).Centroid(2)

notePitch = "G5";

elseif cent < centl(1).Centroid(2) && cent > centl(2).Centroid(2)

notePitch{j} = "E5";

elseif cent < centl(2).Centroid(2) && cent > centl(3).Centroid(2)

notePitch{j} = "C5";

elseif cent < centl(3).Centroid(2) && cent > centl(4).Centroid(2)

notePitch{j} = "A4";

elseif cent < centl(4).Centroid(2) && cent > centl(5).Centroid(2)

notePitch{j} = "F4";

elseif cent < centl(5).Centroid(2) && cent > cent6

notePitch{j} = "D4";

end

pitchStrip{i} = cat(1,notePitch);

end

allPitches = cat(1,pitchStrip);

end

Finally, based on each note’s length and pitch, the notes should be played back accordingly, such that the song can be heard. Since each note value has a relative frequency, these can be mapped. Additionally, the length of the note can also be translated into a specific amount of time played. A wave was constructed based on the frequency and duration for each note. Any component labeled ‘not note’ was ignored. Disappointingly, the method tried out was not productive in playing the notes.

fs = 44100;

for i = 1 : length(allPitches)

for j = 1 : length(allPitches{1,i})

if allLengths{1, i}{1, j} == "quarter"

t = 8;

elseif allLengths{1, i}{1, j} == "half"

t = 4;

elseif allLengths{1, i}{1, j} == "whole"

t = 2;

elseif allLengths{1, i}{1, j} == "not note"

continue

end

if allPitches{1, i}{1, j} == "C4"

p = 261.63;

elseif allPitches{1, i}{1, j} == "D4"

p = 2931.66;

elseif allPitches{1, i}{1, j} == "E4"

p = 329.63;

elseif allPitches{1, i}{1, j} == "F4"

p = 349.23;

elseif allPitches{1, i}{1, j} == "G4"

p = 392.00;

elseif allPitches{1, i}{1, j} == "A4"

p = 440.00;

elseif allPitches{1, i}{1, j} == "B4"

p = 493.88;

elseif allPitches{1, i}{1, j} == "C5"

p = 523.25;

elseif allPitches{1, i}{1, j} == "D5"

p = 587.33;

elseif allPitches{1, i}{1, j} == "E5"

p = 659.25;

elseif allPitches{1, i}{1, j} == "F5"

p = 698.46;

elseif allPitches{1, i}{1, j} == "G5"

p = 782.99;

end

wave = sin(2\*pi\*p\*t);

player = audioplayer(wave,fs);

play(player);

end

end