Presto: An AI-Powered Optical Music Recognition Tool for Musicians

Elisabeth Holm and **Bryant Perkins**
Stanford University
eholm@stanford.edu and bperk25@stanford.edu
GitHub: https://github.com/bperk25/presto

Abstract

This paper covers Presto, a computer-vision-powered sheet music reader designed to accurately interpret and generate musical notation from images of sheet music. The system consists of several stages: image preprocessing, feature extraction, data interpretation, and annotation/generation. Presto combines multiple image processing techniques, including grayscale conversion, Gaussian filtering, and contour detection, with state-of-the-art machine learning models such as YOLOv8 for note recognition. By integrating these methods, Presto offers a comprehensive solution for accurate and efficient sheet music digitization.

1 Introduction

1.1 Motivation

As novice music learners, there is nothing more frustrating than finding sheet music for a piece you want to learn with no playable midi files for reference. It is equally as daunting to have to decipher and manually annotate each note in the composition. Our project streamlines the music learning process for those embarking on their musical journey, supporting individuals with limited music literacy skills and music aficionados alike.

1.2 Significance

Presto represents a significant advancement in music technology. Traditionally, the process of transcribing sheet music into digital formats has been time-consuming and labor-intensive, requiring manual annotation of each note and symbol. This often presents a barrier to entry for individuals with limited music literacy skills, hindering their ability to access and engage with musical compositions effectively.

Presto addresses this challenge by offering a sophisticated computer-vision-powered solution that automates the digitization process with high accuracy and efficiency. This eliminates the laborious manual annotation process, making sheet music more accessible to all music enthusiasts. By integrating advanced image processing techniques and machine learning models like YOLOv8, Presto ensures robust performance across various musical compositions and notation styles. Its user-friendly interface and versatility make it a valuable tool for musicians, educators, composers, and researchers, democratizing access to musical resources and enriching the overall musical experience.

1.3 Challenges

The main challenges we foresaw were identifying staff lines and individual notes, both of which we handled through preprocessing the image to make existing line and blob detection algorithms perform at higher rates of accuracy.

To ensure the project was within scope of the 3-week timeline, our program did make a few simplifying assumptions. We assumed the images would be digital PDFs, not photos of physical copies of sheet music, which would involve challenges of warped and non-uniformly-lit images. We also assumed that the music would only involve individually played notes, such as for a singer or violinist, who can only play one note at a time. This avoided errors due to note blobs touching one another and simplified our interpretation step, but limited the music we could accurately process.

2 Related Work

Optical music recognition (OMR) is a well-established field, with recent tools beginning to use deep learning to more accurately scan and identify musical notation, such as Wel & Ullrich covered in their 2017 paper "Optical Music Recognition with Convolutional Sequence-to-Sequence Models"[1] In addition, though we did not follow her 2012 paper,

[1] https://archives.ismir.net/ismir2017/paper/000069.pdf

it appears that both Presto and Rebelo et al[2] follow a similar 4-step approach to the OMR problem.

However, while sheet music interpreting technology like ours does exist, it is not easily accessible or affordable. Advanced papers do not offer a user-friendly interface for use, and programs such as Neuratron's PhotoScore and NotateMe lock up this amazing technology behind large paywalls (up to 369 dollars[3]) that are intimidating for learners at the beginning of their music journey. There does not currently exist a free, easy-to-use sheet music interpreter that creates both an annotated sheet music document and playable MIDI file, which is where Presto fills the gap. We focused on a simple but powerful design, geared towards accelerating novice music learners with in-sheet annotations and customizable MIDI files to provide material for cross-reference.

3 Methodology

3.1 Preprocess Image

First, we loaded and filtered the png image, creating gray-scale and Gaussian filtered versions using OpenCV's `imread`, `cvtColor`, and `GaussianBlur` (with a kernel of 7x7) functions.

Then, we removed horizontal staff lines and vertical note lines using OpenCV's `threshold`, morphological transformation[4], and `findContours` functions.

For horizontal lines, we thresholded the pixel values, then applied a long (%50 of image width) horizontal kernel and the morphological `open` operation to the thresholded image, isolating long horizontal lines, which we assumed to be staff lines. Then we applied `findContours` to identify the lines' exact locations, creating a threshold image of just staff lines, as well as "erasing" (drawing in white) the staff lines from the original image. This erased anything that intersected with the lines, including parts of notes, so, we used a note-sized kernel (calculated based on image dimensions) and the morphological `close` operation to repair the image, resulting in an image with isolated notes and erased staff lines.

Once we had an image with horizontals removed and notes reconstructed, we used a note-sized kernel and OpenCV's `erode` function to remove vertical note lines, such that notes could later be identified by their isolated blobs.

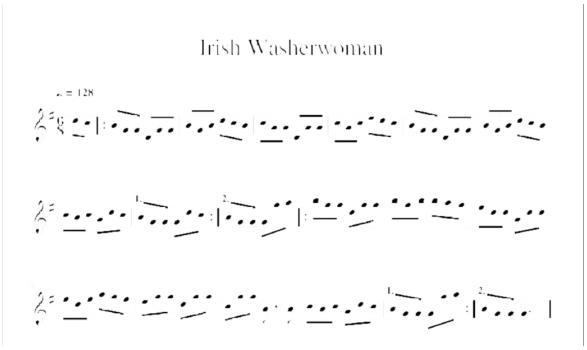

Figure 1: Pre-processed image with horizontal and vertical lines removed

3.2 Find Key Features

Using the isolated blob image from preprocessing, we were able to tune OpenCV's `blobDetector`[5] to find notes. Our custom detector filtered by area, circularity, and inertia in an aim to identify solely notes, and not lyrics or other objects printed on the page. We then created instances of a custom `Note` class that we built, assigning the center of each blob as the x and y coordinates of a new note object.

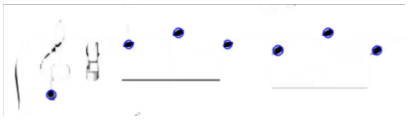

Figure 2: Identified note blobs

For line identification we repurposed our `remove_horizontal_lines` function in order to isolate the staff lines effectively. At first we used a modified version of cv2's Canny edge detector but found that `remove_horizontal_lines` produced more continuous lines. This increased cv2's HoughLineTrasformP performance with identifying the staves.

From these bar lines we were able to collect the average line gap - ignoring outlier gaps between staff sets. We then used this information to remove other possible outliers found by HoughLineTransformP, identify the base staff lines - present right before the remaining outlier gaps, and transfer the final average gap within staff sets to the next stage, interpret.

[2]https://link.springer.com/article/10.1007/s13735-012-0004-6

[3]https://www.neuratron.com/order.asp

[4]https://docs.opencv.org/4.x/d9/d61/tutorial_py _morphological_ops.html

[5]https://learnopencv.com/blob-detection-using-opencv-python-c/

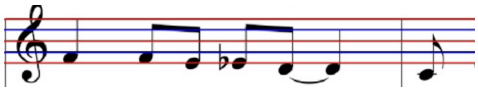

Figure 3: Identified staff lines

3.3 Interpret Data

To identify the "length" or "time" of the note (eg quarter), we trained a YOLOv8[6] image classifier on a music note dataset[7]. Given a cropped image (see fig. 4) of a note, it returns the most likely of 5 classes (eighth, half, quarter, sixteenth, whole), and assigns it to the note.

Figure 4: Sample cropped image of note that is fed to YOLO's classification model. Prediction: quarter note (correct)

Once the Note class objects have their positional (x, y) data, both base staff lines and the average staff line gap can be used to create a dictionary mapping each Note object with their corresponding staff id number. This dictionary is essential in sorting the Notes and assigning their note id property, used to keep the order the notes will be played in. Next we cycle through the base staff lines dictionary and assign y coordinate ranges to each note in the desired key signature.

The final stage is iterating through each of these dictionaries and assigning each Note class object the following properties: staff id, note id, key, and octave. The octave is acquired using the index of the range in the value of the matching note name. Notes that are note located within the range of a staff line set have their note key property set to null. After this process is complete, all non-null Note objects should be fully initialized with all the properties need to produce the MIDI file.

3.4 Annotation & Generation

To create the final annotated sheet music file, we annotate the given original image using `matplotlib`'s plot and text functions to mark each note with its identified key (A, B, ...) and time (eigth, quarter, ...). If the user chooses, they can save this file locally for musical practice.

[6]https://docs.ultralytics.com/quickstart/

[7]https://www.kaggle.com/datasets/kishanj/music-notes-datasets/versions/1/data

The MIDI generation process involves converting the interpreted musical notation into a format that can be played back electronically. This begins with mapping the notes identified in the sheet music to their corresponding MIDI note numbers. Each note is represented by a combination of its pitch and octave. Utilizing a predefined list of notes and octaves, the system translates the symbolic representation of musical notes into numerical values compatible with MIDI standards.

Once the note numbers are generated, a MIDI file is created using the MIDIUtil library. This file comprises a single track, with tempo information added automatically. For each note identified in the sheet music, a corresponding MIDI note event is added to the track, specifying parameters such as pitch, timing, duration, and volume.

Finally, the generated MIDI file is saved to disk, ready for playback using MIDI-compatible software or devices. This process ensures that the digital representation of the sheet music accurately reflects the original musical composition, enabling users to listen to and manipulate the music in a digital environment.

4 Results

4.1 Quantitative Analysis

Before reviewing the statistics, it is important to note that many errors compounded due to the linear nature of our digitization process. We have slightly accounted for this through analysing the stages in good faith, using only the successful outputs for future stage accuracy ratings.

From the graph in figure 5, we observed that

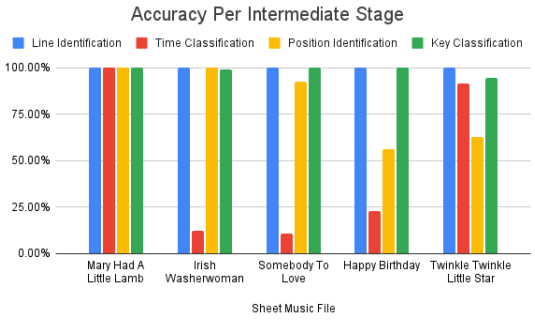

Figure 5: Plotted quantitative performance results for various sheet music files

PRESTO achieved perfect accuracy in line identification for all files, indicating its robustness in detecting and removing staff lines from the sheet

music images. However, there is a noticeable disparity in the accuracy of time classification across different files, ranging from as low as 11.00% to as high as 91.38%. This suggests variability in our training through YOLOV8 and its ability to accurately classify the time of individual notes, which may be influenced by factors such as note density, spacing, and clarity in the input images.

Similarly, position identification demonstrates varying levels of accuracy across files, with percentages ranging from 56.41% to 100.00%. This indicates the system's proficiency in identifying the positions of notes on the staff lines but also highlights potential challenges in cases where note placement is less discernible or where notes are clustered closely together.

In contrast, key classification shows consistently high accuracy rates across all files, with an average accuracy of 98.78%. This is most likely due to its more direct dependency on the base staff lines in order to predict the note key properties. Because our base staff lines are found with 100% accuracy, it is highly likely that the key properties will be correctly identified.

4.2 Qualitative Analysis

Throughout our experiments we noticed a few sheet music properties that would soon be realized as errors throughout our digitization process. Namely insufficient quality, note cluster discrepancy, note center slippage, and incompatable note complexities.

Throughout lower quality files with higher ranges of pixel intensity values, we noted a high loss of significant data during our remove horizontal lines process. This would result in lines being lost and HoughLineTransformP returning zero lines.

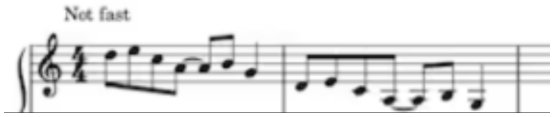

Figure 6: A low-quality image where no lines were detected

In more complex files, containing stacked notes and chords, we noticed a pattern of cv2's BlobDetector only returning one position - between two notes. This would result in a completely incorrect note being added with the loss of two. Another issue observed was the slight slippage of note centers

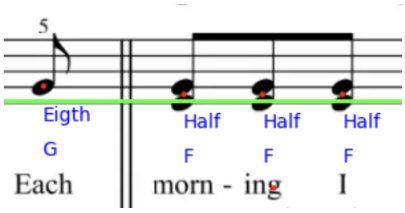

Figure 7: Two concurrent notes being identified as one due to image preprocessing and note reconstruction

outside of the desired range for the correct note key. This was due to the nature of the notes' ellipses shape and the BlobDetector's requirements for circle similarity. This conflict of interest would result in the center of the note heads being identified as slightly below or above their actual center. Lastly

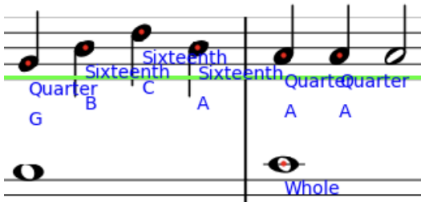

Figure 8: Identified center of note heads (red) is slightly above their actual center

the note time classification saw a heavy decrease in performance when the complexity of the note format was increased. This included notes that were not standalone but instead tied, stacked, dotted, or connected/barred together.

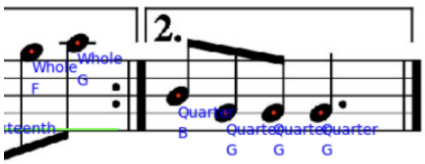

Figure 9: Grouped eighth notes, predicted to be quarter notes, and a true quarter note

Despite all of these observed errors we saw high rates of success regarding note key classification and line identification.

5 Conclusion

5.1 Summary of Findings

Based on the quantitative and qualitative analysis presented, several key insights emerge regarding

the performance of PRESTO in digitizing sheet music.

It is evident that PRESTO achieves great reliability in line identification, leading to near perfect annotation abilities for note. However, there are notable discrepancies in the accuracy of time classification and position identification, with varying levels of success observed across different sheet music files. The variability in time classification accuracy is most like due to the limited training data, as all of the development and testing images were standalone notes. Similarly, challenges in position identification highlight potential issues with discerning note placement, especially in cases of clustered or densely packed notation. This could be due to the inherit limitations in cv2's BlobDetectors as it is outside of it intended use with more complex figures like the different note structures.

Overall PRESTO produced accurate note key annotations with MIDI files that are in tune but slightly out of time.

5.2 Implications

The success of the PRESTO program holds significant implications for both novice music learners and seasoned musicians alike. Accessibility and affordability were key (pun intended) considerations, with existing solutions often inaccessible, and PRESTO could eliminate financial barriers and empower users to engage with sheet music more effectively. Overall, PRESTO holds potential as a significant advancement in music education and accessibility, encouraging more inclusive, interactive, and enjoyable music learning for all.

5.3 Potential Future Work

In the future, we'd like to address challenging cases like identifying whole notes and removing clefs. We also would like to automate key identification, as we currently assume the user can identify and input the key. Beyond digital sheet music, we'd like to broaden compatibility to include printed or handwritten sheet music, which poses challenges due to inconsistent lighting and image warping due to camera tilt. Lastly, though we excel on music meant for singers and single-note instruments, we aim to expand to chord-based music such as piano scores such that a broader range of musicians can find great value in Presto.

6 Individual Contributions

Elisabeth Holm worked on image preprocessing (including horiztonal and vertical line removal), note identification, and YOLO classification of individual notes to identify their length.

Bryant Perkins worked on staff line identification and interpretation, interpreting note-to-staff relationships, playable MIDI file generation, and gathering statistical performance data.

Both worked equally to create the presentation and report.