Discotective

An Optical Music Recognition System

Katie Bouman, Brad Campbell, Mike Hand, Tyler Johnson, and Joe Kurleto

4/25/2011

I. INTRODUCTION

The Discotective is an Optical Music Recognition (OMR) system. This is similar to an OCR system except that instead of recognizing text, it reads musical scores. OMR systems take in an image of a sheet of music, apply image processing algorithms, and output the captured song as an audio signal.

Our main motivation was to help beginning musicians to read sheet music. Learning to read musical notation can be difficult, especially when the reader does not know how the music should sound. A possible solution to this problem is an OMR system that converts sheet music into its corresponding sounds. The Discotective allows a user to quickly get a sense for the music they are learning, helping him or her to learn more effectively. A secondary commercial value is the possibility for digital archival of music. Digitally storing music in a MIDI file for example, as opposed to an image scan, requires less memory and allows users to play the song or even make edits.

After image acquisition, the Discotective utilizes image processing techniques such as filtering, matrix transforms, and classification algorithms to recognize and identify the notes and symbols of sheet music. These classified notes and symbols are output as audio output using direct digital synthesis.

II. PROJECT DESCRIPTION

2.1 System Algorithm Description

The Discotective music transcription process consists of the following steps:

1. Image Acquisition
2. Image Preprocessing
3. Segmentation
4. Classification
5. Musical Output

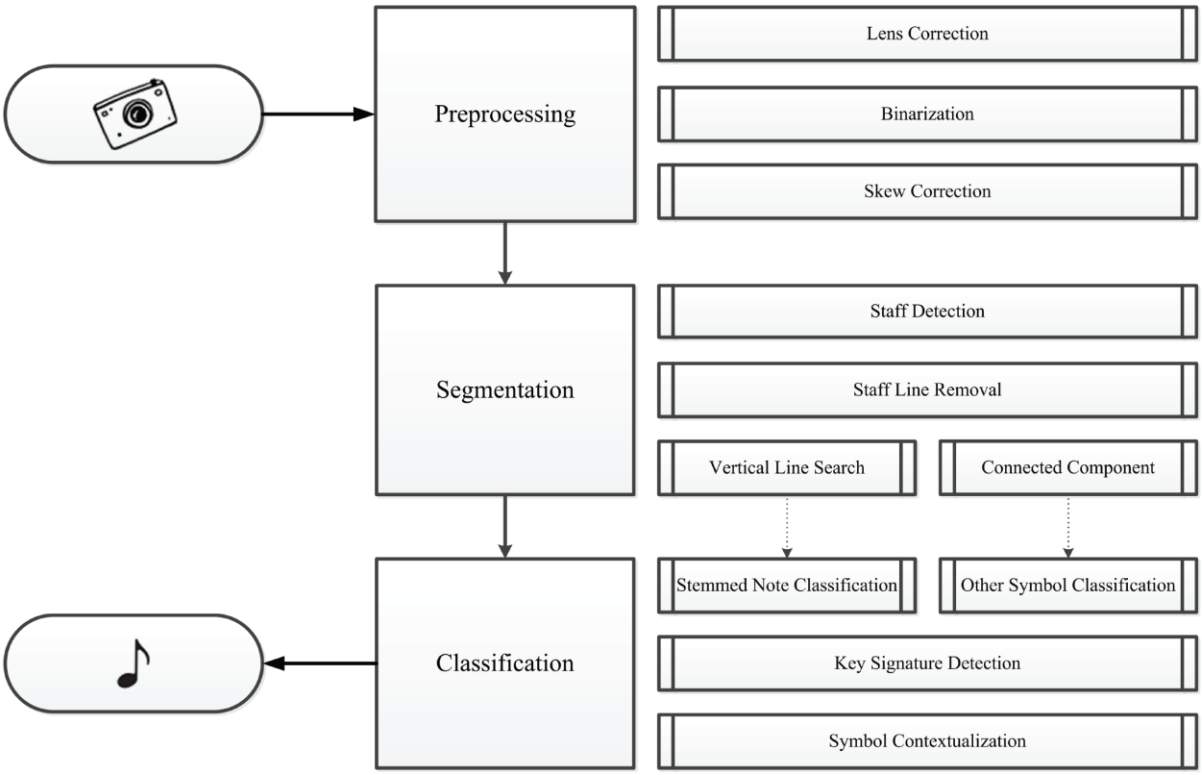

Figure 1: System Overview

The steps are described in greater detail starting on page 4.

Image Acquisition

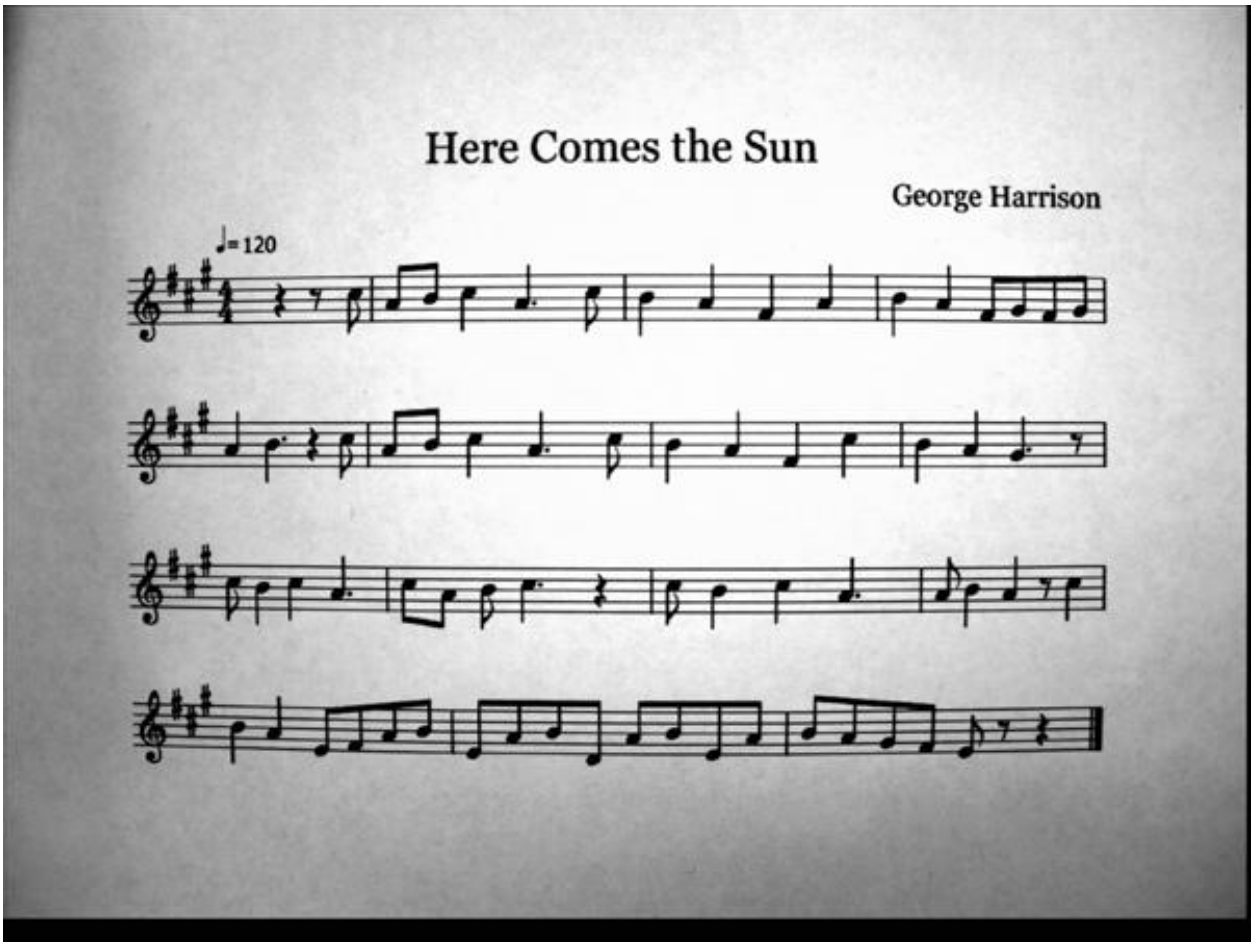

Figure 2: Unprocessed Grayscale Image

The first step in transcribing sheet music is capturing an image of the music. To accomplish this, we used an Altera D5M camera interfaced it with an Altera DE2 FPGA board. Altera supplies sample interface code served as the starting point for using the camera. We continuously captured frames from the camera and store it in the SDRAM on the FPGA. Each frame from the camera is a five-megapixel image in RGB Bayer Pattern format, which is converted to a grayscale image in the hardware using bilinear interpolation demosaicing. This image is output continuously to the VGA Monitor until the user stops the camera. Once the camera is stopped, it no longer writes incoming data to the SDRAM. The most recent image stored in memory is passed on to be used in the following stages of the process. At this point, the image is ready to be processed.

Image Preprocessing

In image preprocessing, the software takes over from the hardware. It converts the grayscale image to a binary image, corrects for lens distortion, trims the image, segments the image into staffs, and determines the key signature.

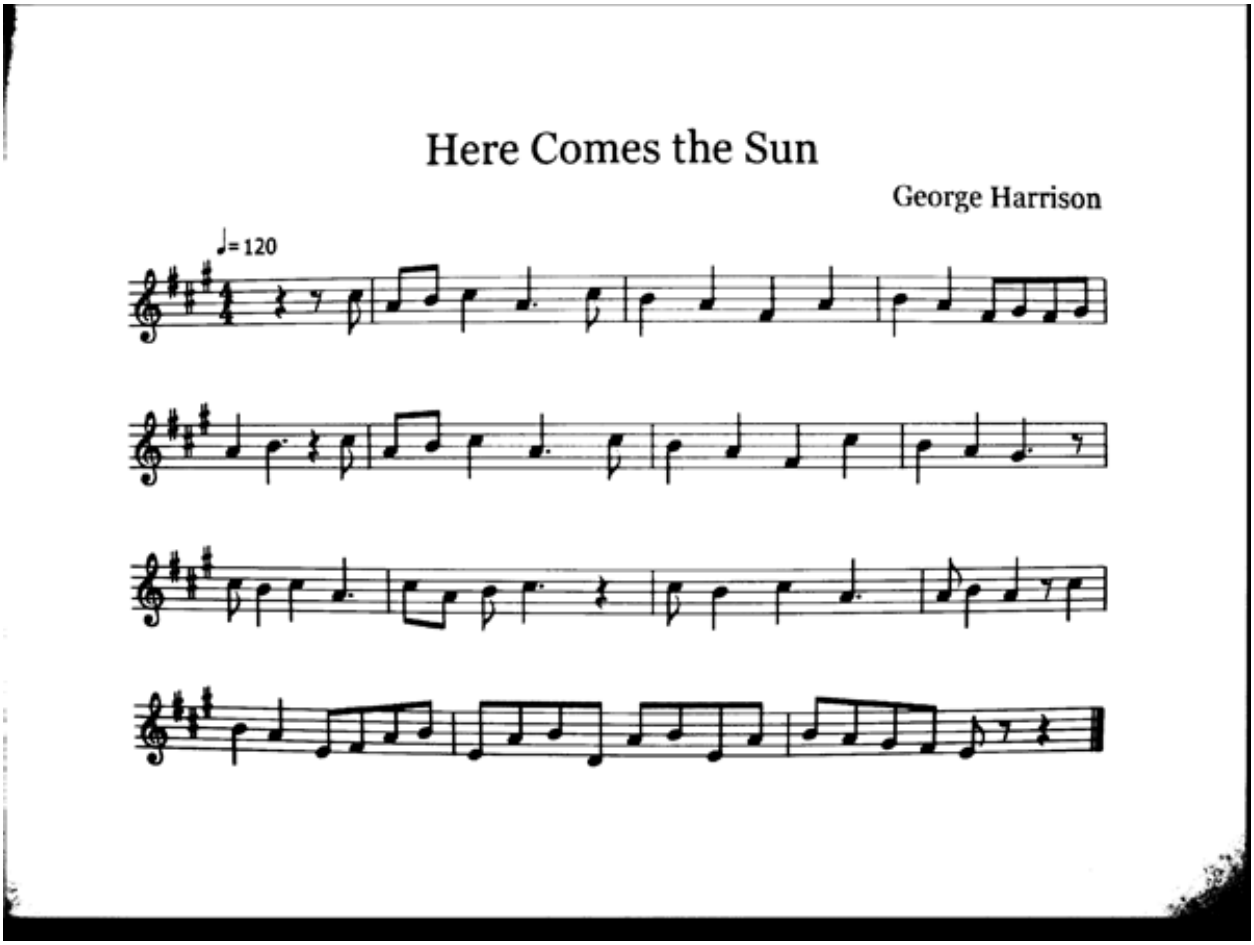

Figure 3: Binary Image

The software converts the grayscale image to a binary image. Using an adaptive threshold, the software determines whether each pixel in the image should be white or black and assigns the value accordingly. The adaptive filter is determined using a quadratic linear regression algorithm. Using a binary image takes eight times less memory space than a grayscale image and allows simpler processing. Unfortunately, this adaptive binarization algorithm is too computationally expensive to be used on the FPGA. Thus, in our hardware implementation of the algorithm, a static threshold is used to binarize the image.

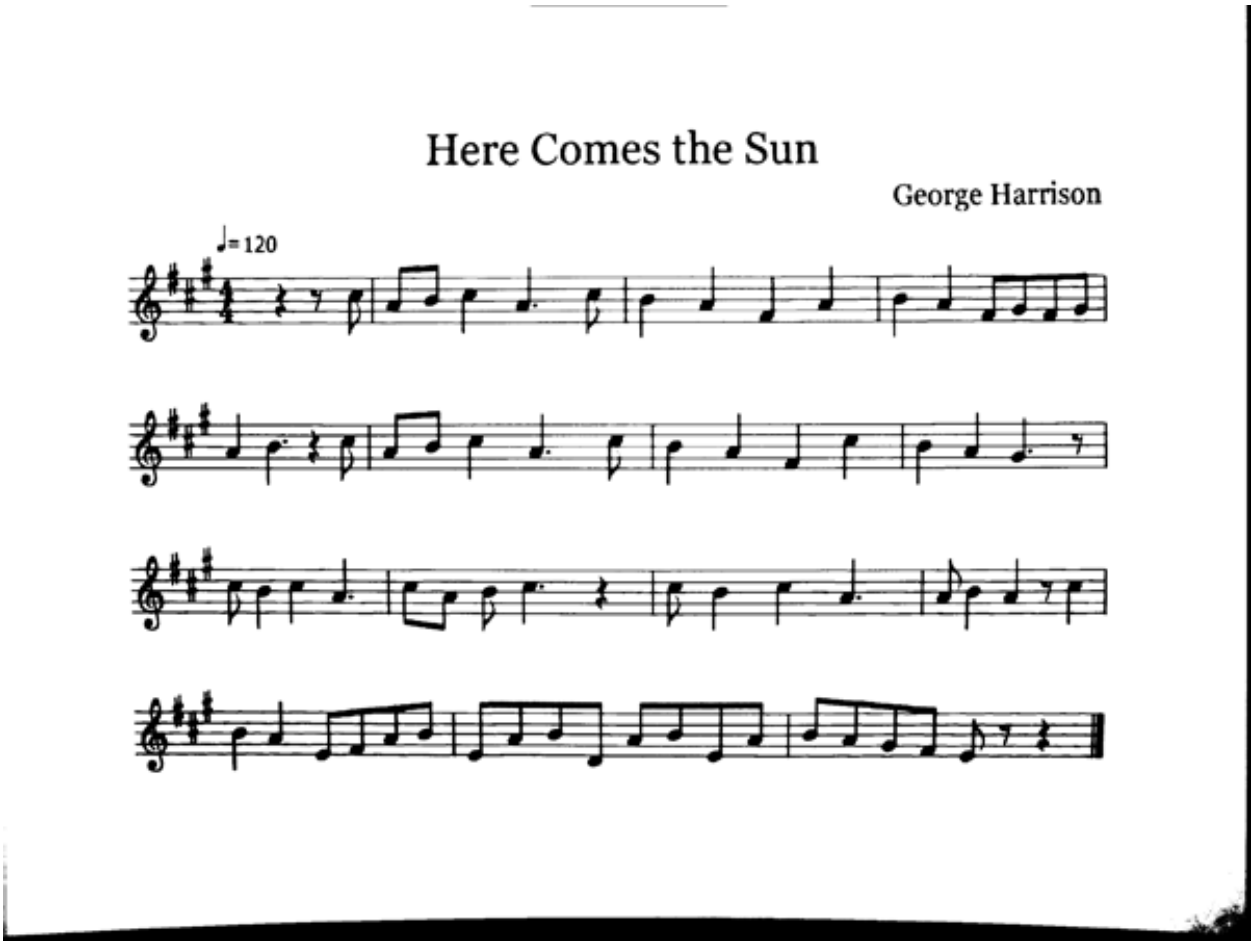

Figure 4: Binary Image Free of Distortion

The next step of the algorithm is to correct for lens distortion. The fisheye distortion applied by the camera is removed by using a parabolic transformation which maps pixel data to different indices. This reliably reconstructs the image free of distortion.

After the image has been binarized and corrected for fisheye we correct for any rotation or skew due to camera placement. While capturing the image, the plane of the camera's sensor will rarely be parallel to the sheet of music. Additionally, the camera may be rotated so that the top of the image is not parallel to the staff. This camera misalignment will distort the image's proportions, and correcting for rotation and skew greatly simplifies the classification of notes. This functionality was not used in the algorithm developed for the FPGA due to its high computational complexity. Thus, all images passed to the algorithm on the board were assumed to be aligned with no skew or rotation.

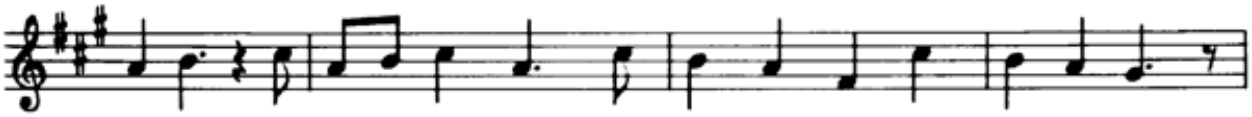

Figure 5: A Single Staff of Music

The software then segments the image into staffs. It uses a projection onto the y axis to determine the location of the staff lines. Staff lines run almost the entire length of the image and their projections are larger than surrounding data by a significant margin. These staff lines are then grouped into groups of five. From the data contained in these groups, the image is divided into its different staffs.

The software determines the key signature by using projections on the x axis to locate sharp and flat symbols. Key signatures never contain a mix of sharps and flats. As such, when both sharps and flats are located, the software determines them all to be whichever symbol had more occurrences. From the number of sharps or flats that are found, the key signature is determined.

The staffs are then sent one by one to the later steps of the process

Segmentation

Segmentation consists of removing the staff lines, and isolating the notes with stems.

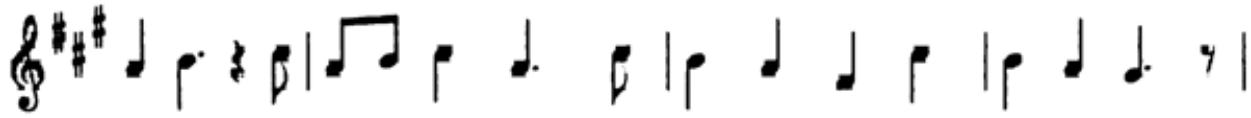

Figure 6: A Staff Free of Staff Lines

The software removes the staff lines from the image by whiting out pixels where staff lines are known to be located. As the staff lines are being removed, the pixels above and below the image are checked to determine if the staff line intersects a symbol. If the staff line intersects a symbol, that portion of the line is not removed.

Figure 7: Stemmed Note Segmentation

The software isolates notes with stems by taking vertical projections of the staff image. Peaks in the image correspond to either measure markers or notes with stems. Once these are located, the vertical lines are removed and the characteristics of the note or measure marker are saved to be used later in classification.

Figure 8: Stemmed Note Removal

Classification

Classification consists of assigning pitch and duration values to stemmed notes, and locating and characterizing whole notes.

The software assigns pitch and duration values to stemmed notes based on their characteristics. Pitch is determined based on the center of mass of the note head. Whichever staff line or space is closest to the center of mass of the note is assumed to be the intended note. Duration is determined by checking if the note head is filled or open and if it has a tail. If the note head is open, the note is a half note. If the note has a tail it is an eighth note. Otherwise it is a quarter note.

Whole notes are located later in the classification process. After the symbol is located, the symbol's features are compared against a set of known characteristics for whole notes. If it is found to be a whole note, its duration is assigned accordingly, and its pitch is determined from its center of mass.

Musical Output

Musical output consists of using the previous pitch, duration, and key signature data to output the corresponding music. The software takes all of the pitch and duration data from classification, along with the data for the key signature and assembles it into a list of frequency and time values. The software then uses these values in conjunction with direct digital synthesis and the FPGA's audio codec to output the waveforms corresponding to the sheet music. Additional tonal qualities were achieved by adding harmonic frequencies for each note.

2.2 System Architecture

The Discotective is built on an Altera DE2 FPGA. A 50 MHz softcore processor has been downloaded to the FPGA, making the system compatible with C programs. In addition, a 5 megapixel camera from Terasic captures images, which are used for processing and can be displayed on a VGA monitor. The final audio is output to speakers via the board's audio codec.

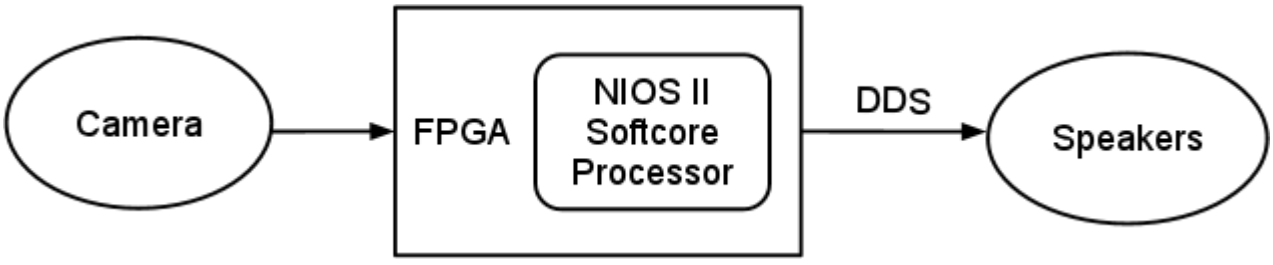

Figure 9: Hardware Overview

2.3 Parts List

Our project made use of the following hardware components:

- Altera D5M Camera (Terasic), $85
  http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=68&No=281
- Altera DE2 FPGA Board

III. MILESTONES

3.1 Milestone 1: March 17

Goal:

- Algorithms designed and written in C
- Familiarization with hardware

Reality:

- MATLAB prototype missing final classification algorithms
- C implementation underway
- Able to acquire and store images with hardware

3.2 Milestone 2: April 7

Goal:

- Full implementation running on the hardware

Reality:

- Full implementation in C on Linux
- Some portions running on the FPGA

Since we did not meet all of our first milestone's goals, we decided to adjust our second milestone's goal to have just a full implementation in C. This meant we had to spend more time before the expo porting our code to the board and debugging. Overall we had made our goals ambitious to push us to work hard.

IV. DEMONSTRATION OF PROJECT

We demonstrated the Discotective at both the Design Expo and our final presentation. At the Design Expo, our hardware implementation was working with basic music samples, and we had "Happy Birthday" and a simplified version of "The Victors" available at the table. Visitors were able to approach the Discotective, take an image of the music, and watch the results of our algorithms in real time. To add variety and display certain details of our algorithms, we created a graphical user interface for our MATLAB prototype, which we set up next to the hardware

implementation. This MATLAB version was capable of running up to twenty songs, which were preloaded on our computer.

By our final presentation, we had fixed many of the errors in our FPGA implementation, and we were able to demonstrate more complicated music. The Discotective performed three songs: the classic Tetris theme song, "Saria's Song" from the Legend of Zelda: Ocarina of Time, and The Beatles' "Here Comes the Sun."

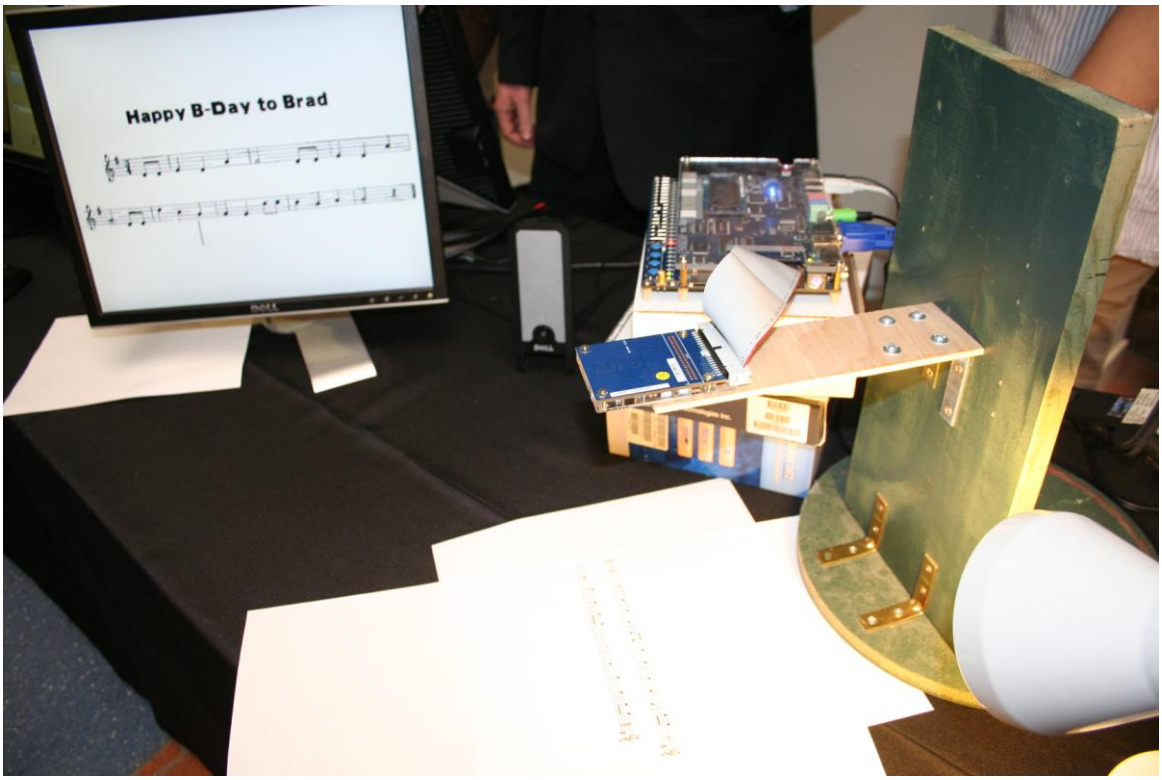

Figure 10: Design Expo Set-Up

Overall, the Discotective performs well with simple music, such as that played by a middle school band. Due to the limits of our design, however, many symbols are difficult to classify. Slurs or ties connected to notes, for example, cannot be segmented properly with connected-component analysis. More complicated symbols, such as chords, repeat signs, accents, or text will not be properly recognized by the classifier. The hardware implementation constraint introduces further limitations. The slow softcore processor prohibits the use of more advanced binarization and deskew preprocessing steps, which reduce the accuracy of later algorithms. Nevertheless, the device still performs well under proper conditions.

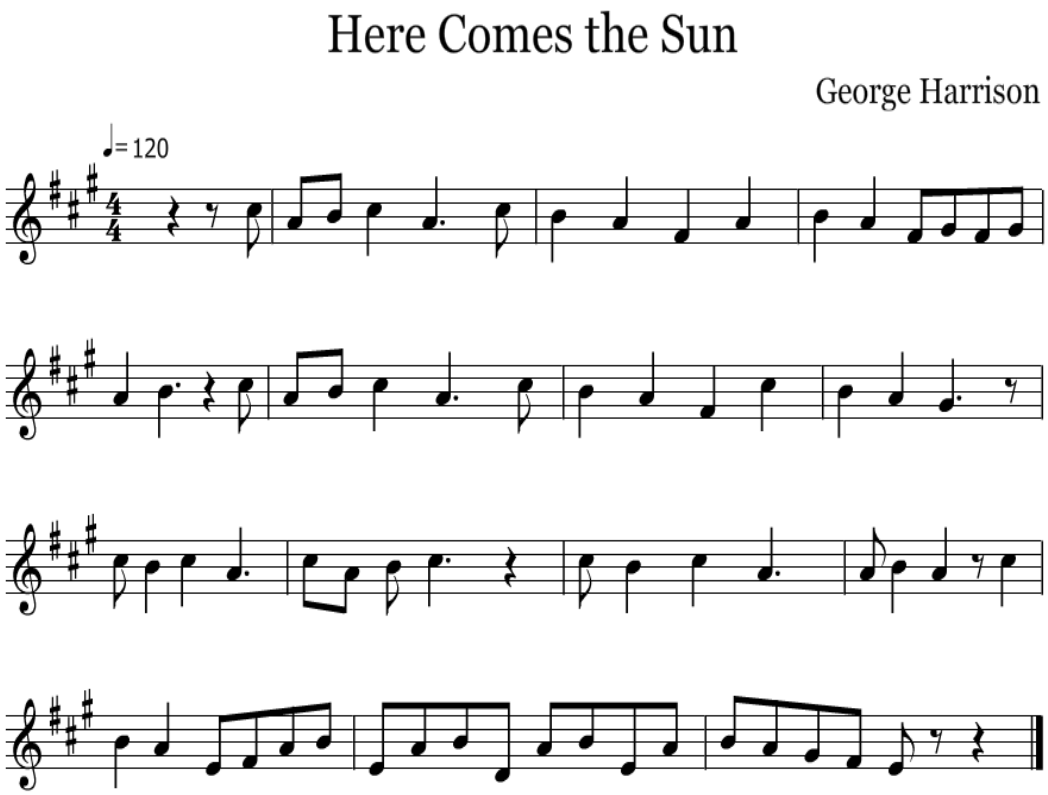

Figure 11: Sheet Music Used at Presentation

V. Team Member Contributions

Team Member	Contribution	Effort
Katie Bouman	MATLAB Algorithms, C code conversion	19%
Brad Campbell	Hardware Design, Camera Interfacing, Low-Level Software	24%
Michael Hand	C code API Design, C code Conversion	19%
Tyler Johnson	MATLAB Algorithms, test cases, poster, presentation slides	19%
Joe Kurleto	MATLAB Algorithms, test cases, poster, presentation slides	19%

VI. References

Bainbridge, David and Tim Bell. "The Challenge of Optical Music Recognition." Computers and the Humanities 35 (2001): 95-121.

Duda, Richard O., David G. Stork and Peter E. Hart. Pattern Classification and Scene Analysis. New York: Wiley, 2000.

"Terasic - Daughter Cards - Video & Image - 5 Mega Pixel Digital Camera Package." N.p., 06 Sep 2010. Web. 20 Apr 2011. <http://www.terasic.com/downloads/cd-rom/d5m/THDB_D5M_CD_v.1.2.0.rar>.

Zhang, Rong. "Image Rectification: Remove Projective and Affine Distortions." Purdue.edu. 9 February 2011 <http://cobweb.ecn.purdue.edu/~kak/courses-i-teach/ECE661.08/solution/hw2_s2.pdf>.