

이미지 처리와 얼룩 검출을 통한 점자 악보 인식과 MIDI 변환

손용락, 권혁진, 주성진 박세준, 김영종

Braille Music Recognition and MIDI Conversion Using Image Processing and Blob Detection

Yong-Lak Son, Hyeok-Jin Kwon, Seong-Jin Joo, Se-Jun Park, Young-Jong Kim

요약

점자 악보는 시각 장애인들을 위한 악보이며, 6점식 점자 체계를 사용하고 옥타브, 코드, 쉼표 및 다양한 음악 기호들이 점자로 변환되어 읽어되어 있다. 점자 악보는 기존의 점자 체계와는 다른 독자적인 체계를 사용하기 때문에 시각 장애인들이 이를 기억하는 것은 굉장히 어렵다. 따라서 다수의 시각 장애인들은 점자 악보를 파악하는 데 있어서 어려움을 겪고 있다. 본 논문의 목적은 이런 어려움을 겪는 시각장애인들이 수월하게 점자 악보를 인식할 수 있도록 점자 악보 변환기를 만드는 것이다.

본 논문에서는, OpenCV의 기본 모듈을 활용하여 점자 악보를 인식하고 MIDI 파일로 변환하는 방법을 제시한다. 이미지 처리 과정으로 이미지 자르기, 회색조, 적응형 임계 처리, 가우시안 블러, 이미지 임계 처리를 사용했고, 얼룩 검출로는 OpenCV의 SimpleBlobDetector 모듈을 사용했다. 그 결과 성공적으로 점자를 인식하였고, 인식한 정보를 바탕으로 MIDI 파일을 만들어 재생하는데 성공했다.

ABSTRACT

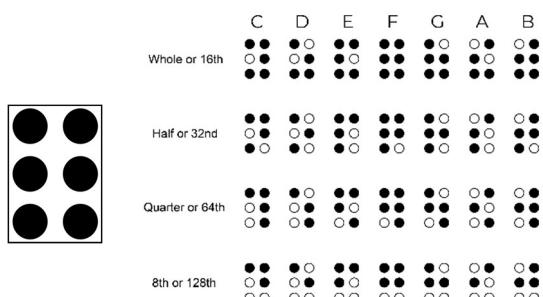
Braille music score is designed for blind, which is composed with six embossed points, and it gives information about octave, code, rest, and other music symbols. Braille music score use its own standard which is different with braille alphabet, so it's very hard for blind to remember this. Because of this, many blinds have problem with using a braille music score. The purpose of this paper is to provide the braille music converter to give helps to blind to recognize the braille music easily.

In this paper, we propose a method to convert the braille music score to MIDI file using OpenCV default module. For image processing, we use Cropping, Grayscale, Adaptive thresholding, Gaussian blur, Image thresholding. For blob detecting, we use OpenCV's SimpleBlobDetector. As a result, we successfully recognize the braille score and built the MIDI file based on the information we've made.

Key words- Image Processing, Blob Detection, OpenCV, MIDI Conversion

I. 서 론

앞을 보지 못하는 시각 장애인은 많은 것을 소리에 의지한다. 그렇기에 그들은 음악에 관심이 생기는 경우가 많다. 그들은 음악을 배우기 위해서 점자 악보를 사용한다. 점자 악보는 문자 점자와 마찬가지로 6 점식 점자를 사용하며, 악보에 존재하는 음악 기호들에 관한 정보를 포함하고 있다. 음표, 박자, 가사, 빠르기, 도돌이표, 세기 등등과 같은 다양한 음악 정보들을 나타낼 수 있으며, 앞이 보이지 않는 시각 장애인들이 음악을 배울 수 있도록 도와준다. 그림 1은 6 점식 점자와 점자 악보에서 사용하는 음표의 모습이다.



[그림 1] 점자 악보 기호

점자 악보는 6 점식 점자를 사용한다는 점에서는 문자 점자와 동일하지만, 그것과는 완전히 다른 체계를 갖고 있다. 따라서 문자 점자에 익숙한 시각 장애인도 점자 악보를 통해 음악을 배우기 위해서는 새로운 체계를 공부하고 기억해야 한다. 악보의 기호는 굉장히 다양하기 때문에 점자 악보를 읽기 위해서는 다양한 패턴을 숙지하고 암기해야 한다. 이 과정은 굉장히 힘들고, 많은 시간을 요구한다. 따라서 시각 장애인들은 음악을 배우는 데 있어서 큰 어려움을 겪고 있다.

점자 악보를 다루기 위해서 도움을 받는 것도 쉽지 않다. 국내에는 점자 악보 관련

정보가 충분하지 않다. 점자 악보를 생산하는 곳도 한정적이고, 자료도 일반 악보에 비하면 터무니없이 적다. 그러므로 관련된 시설이나 서비스가 부족하고, 점자 악보를 잘 아는 전문가의 수도 많지 않다. 따라서 시각 장애인이 점자 악보와 관련된 도움을 받기는 힘들다.

우리는, 소프트웨어 프로그램을 통해서 그들에게 도움을 줄 수 있는 방법을 찾고자 연구를 진행했다. 점자 악보를 인식하여 재생 가능한 파일로 변환하고, 악보로 바꾸어 보여준다면 음악에 대해서는 잘 알지만, 점자에 대해서는 잘 모르는 사람도 변환된 정보를 바탕으로 시각 장애인의 학습을 도와줄 수 있고, 시각 장애인도 촉각과 청각을 모두 활용하여 학습을 진행할 수 있기에 기존보다 더욱 빠른 점자 악보 학습이 가능해진다.

본 논문에서는 이를 위해서 점자 악보를 인식하고, 이를 재생 가능한 MIDI 파일로 변환시키는 방법을 제안한다. 이를 위해 기존의 문자 점자 인식 방법과 유사한 방식을 사용하여 점자를 인식하고, 점자 악보의 규칙을 파악하고 데이터베이스화하여 추출한 점자 정보에 해당하는 악보 관련 정보로 변환하는 방법을 제시한다.

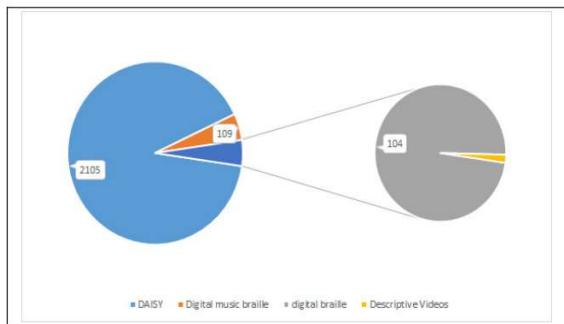
본 논문의 구성은 다음과 같다. 제2장에서는 점자 악보를 인식하고, 이를 MIDI로 변환하기 위해서 사용한 방법을 소개한다. 과정에는 이미지 처리, 점자 인식, MIDI 변환이 포함된다. 제3장에서는 “할아버지 낡은 시계” 점자 악보를 사용하여 실제 점자 악보 인식 및 MIDI 변환 실험을 진행한다. 그리고 그 결과를 분석한 뒤, 의의 및 한계에 대해서 토론한다. 제4장에서는 결론 및 향후 연구 방향을 제시한다.

II. 관련 연구

| year | Total |
|-----------|-------------|
| 2016 | 1,863(+66) |
| 2015 | 1,797(+101) |
| 2014 | 1,696(+201) |
| 2003-2013 | 1,495 |

(Source: The National Library of the Disabled)

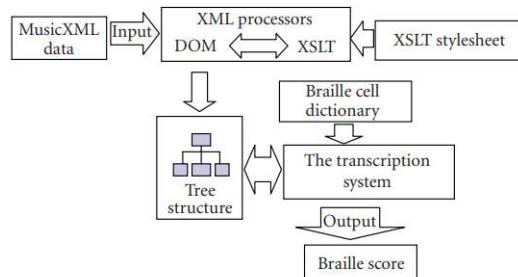
[그림 2] 점자 악보 총량



[그림 3] 대체 자료 신청 현황

그림 2 는 국립 장애인 도서관에서 보유하고 있는 전자 점자 악보의 총량을 나타내고, 그림 3 은 2016 년 한 해 동안 국립 장애인 도서관에서 발생한 대체 자료의 신청 현황을 나타낸다. 그림 2 를 통해서 전자 점자 악보의 증가율은 높지 않은 것을 알 수 있고, 그림 3 을 통해서 점자 악보 자료의 신청 현황은 미디어 자료(DAISY)에 비해 굉장히 낮은 것을 알 수 있다. 이는 점자 악보의 자료 숫자가 적고, 점자 악보의 학습이 어렵기 때문이다. 따라서 이 두 자료로, 점자 악보의 생태계는 상당히 열악하고, 잘 활용되지 않는 모습을 확인할 수 있다.

지금까지의 연구는, 점자 악보 자료의 수를 늘리기 위해, 주로 일반 악보를 점자 악보로 변환하는 것에 대해 다루었다. 그림 4 는 이러한 연구의 예시이고, 악보를 점자 악보로 변환하는 과정의 흐름도이다. 해당 연구에서는 MusicXML 의 구조를 분석하고, 이를 tree 형태의 점자 악보 구조로 변환한다. 그리고 이를 표준 형식을 따라 점자 악보로 제작한다.



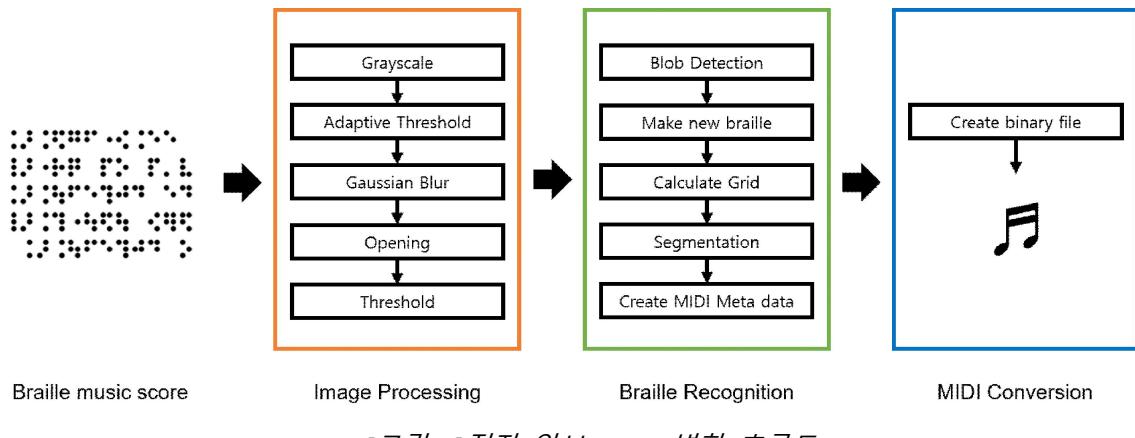
[그림 4] XML 점자 악보 변환 흐름도

이외에도 점자 악보의 생태계를 활발하게 하기 위해서 일반 악보를 점자 악보로 변환하는 연구는 다양하다.

하지만 이와 반대로, 점자 악보를 일반 악보로 변환하는 연구는 드물다. 학습이 어려운 점자 악보의 특징 상 자료가 많아져도 접근성이 떨어지기에 점자 악보 생태계 활성화하기에는 어려움이 존재한다. 따라서 본 논문에서는 이러한 한계점을 극복하기 위해 일반 악보에서 점자 악보로 변환하는 과정의 역순인 점자 악보를 일반 악보로 변환하는 방법을 제안한다. 제안된 방법을 사용한다면, 미디어 파일이 아닌 점자 악보만으로도 청음을 통한 음악 교육이 가능하다. 또한 점자에 대해 잘 모르는 비시각장애인도 변환된 악보를 통해 시각장애인의 학습을 도울 수 있다. 이러한 기능들은 점자 악보의 생태계를 활발하게 만들 뿐만 아니라 시각장애인의 음악 교육에 효과적이다.

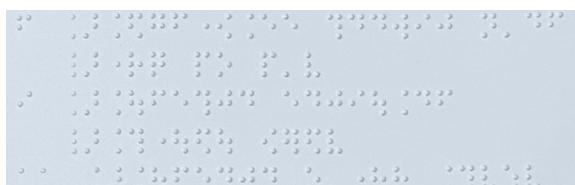
III. 제안 방법

점자 악보를 인식하고, 이를 MIDI로 변환하기 위해서 본 논문에서는 크게 3가지 단계를 제시한다. 첫 번째로는 이미지 처리 단계이고, 두 번째로는 점자 인식 단계이고, 마지막으로는 MIDI 변환 단계이다. 처음 입력 값은 점자 악보 사진이고, 최종 결과물은 MIDI 파일이 된다. 그림 5는 점자 인식 및 변환 과정의 전체적인 흐름도이다.



3.1 이미지 처리

이미지를 올바르게 인식하기 위해서는 처리 과정이 필요하다. 인식 모듈에 이미지를 넣기 전에 데이터가 처리 되어야 올바른 결과를 얻을 수 있다. 이미지를 가공하기 위해서 Grayscale, Adaptive threshold, Gaussian blur, Opening, Threshold 를 사용하고 그 결과 노이즈가 제거되고 원하는 오브젝트만이 남은 이진화 된 이미지를 얻는다. 이미지 처리 과정은 OpenCV 의 모듈들을 활용한다. 그림 6은 스마트폰으로 촬영한 점자 악보이다.



[그림 6] 이미지 원본

3.1.1 회색조

Grayscale 을 적용하면 이미지는 RGB 색상 값을 Grayscale 값으로 변화하여 밝기만을 표현하게 된다. 이미지를 흑백으로 바꾸어준다. Grayscale 은 다음의 공식을 이용해서 사진을 Grayscale 로 변환한다.

$$RGB [A] \text{ to Gray: } Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

R 은 red 영역, G 는 Green 영역, B 는 blue 영역의 색상 값을 나타낸다. OpenCV 에서 다음의 모듈을 사용하면 된다.

`cvtColor(src, bwsr, cv::COLOR_RGB2GRAY)`

그림 7은 grayscale 을 적용한 모습이다.



[그림 7] 회색조

3.1.2 적응형 임계값 처리

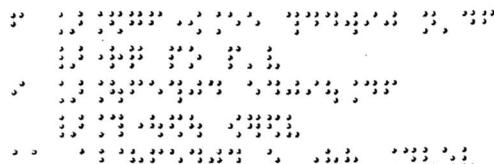
원하는 물체의 모양을 좀 더 정확히 판단하기 위해서는 배경과 물체를 구분해야 한다. 이를 위해 이미지 이진화를 진행한다. 본 논문에서는 adaptive threshold 를 사용한다. adaptive threshold 는 임계값이 가변이며, 픽셀 위치마다 서로 다른 임계값을 계산하여 사용한다. 고정 임계값을 사용한다면 밝기 변화가 심한 이미지의 경우 물체와 배경이 명확하게 구분되지 않는 단점이 있는데, 가변 임계값을 사용하면 해당 단점을 보완할 수 있다. 사용하는 수식은 다음과 같다.

$$T(x, y) = \frac{1}{n^2} \sum_{x_i} \sum_{y_i} I(x + x_i, y + y_i) - C$$

OpenCV에서 다음의 모듈을 사용하면 된다.

```
adaptiveThreshold(img, value, adaptiveMethod,
                  thresholdType, blockSize, C)
```

그림 8 은 adaptive threshold 를 사용하여 이진화를 진행한 모습이다. 물체와 배경이 명확하게 구분된 것을 확인할 수 있다.



[그림 8] 적용형 임계값 처리

3.1.3 가우시안 블러

Gaussian blur 는 gaussian function 을 이용한 kernel 을 사용하여 blur 를 하는 방법이다. 2D Gaussian 은 다음과 같다.

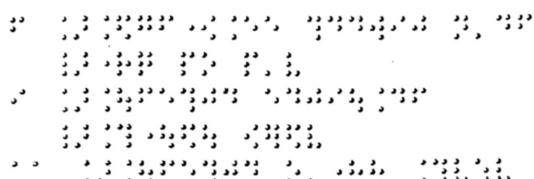
$$G_0(x, y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}}$$

μ is mean, σ^2 is variance

Gaussian blur 는 많은 노이즈가 있는 이미지를 가공하는데 있어 좋은 방법이다. 정보의 손실을 최소화 하면서 가능한 많은 노이즈를 제거할 수 있다. 따라서 인식의 정확성을 높이기 위해서 해당 방법을 사용한다. OpenCV 에서 제공하는 gaussian blur 모듈의 사용법은 다음과 같다.

```
GaussianBlur(src, dst, Size(w, h), sigmaX, sigmaY)
Size(w, h) is size of kernel, sigma is Standard deviation
```

그림 9 는 gaussian blur 를 적용한 모습이다. 점자의 경계선이 부드러워진 것을 확인할 수 있다.



[그림 9] 가우시안 블러

3.1.4 침식 및 확장 연산

Erode 연산은 침식 연산으로, 정사각형 혹은 원의 형태로 저장된 kernel 을 합성한다. Kernel 이 이미지 위에서 연산이 될 때 겹치는 최소 픽셀 값을 계산하고, 기준점의 이미지 픽셀 값을 최소 값으로 교체한다. 연산 식은 다음과 같다.

$$\text{dst}(x, y) = \min_{(x', y'): element(x', y') \neq 0} \text{src}(x + x', y + y')$$

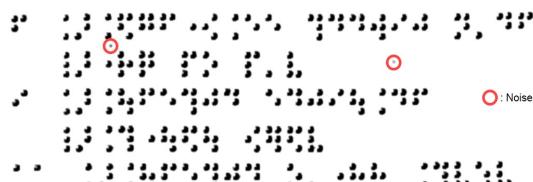
Dilate 연산은 확장 연산이라고 부르며, erode 와 마찬가지로 kernel 을 합성하지만 기준점의 이미지 픽셀 값을 최대 값으로 교체한다. 연산 식은 다음과 같다.

$$\text{dst}(x, y) = \max_{(x', y'): element(x', y') \neq 0} \text{src}(x + x', y + y')$$

Opening 은 erode 와 dilate 연산을 순서대로 수행하는 연산이다. 이진화 된 이미지에 대해서 erode 를 먼저 적용하고, dilate 를 적용한다. 연산을 통해 미세한 개체들을 제거하고 유효한 이미지를 얻어낼 수 있다. OpenCV 에서 다음과 같이 사용하면 된다.

```
erode(src, dst, kernel, anchor, iterations, type, value)
dilate(src, dst, kernel, anchor, iterations, type, value)
```

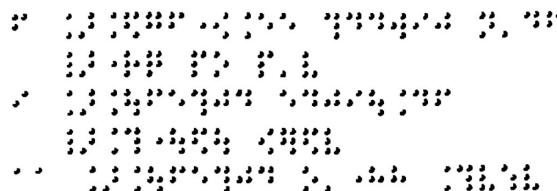
그림 10 은 opening 을 적용한 모습이다. 유효한 점자 마크를 얻는 것에는 성공했지만, 아직 약간의 노이즈가 남아있는 모습을 확인할 수 있다. 해당 노이즈는 threshold 를 사용하여 제거한다.



[그림 10] Opening

3.1.5 임계값 처리

남아있는 노이즈는 threshold 를 사용하여 제거한다. 여기에서는 고정 임계값을 사용한다. 그림 11 은 threshold 를 진행하고 난 뒤 노이즈가 제거된 모습이다.



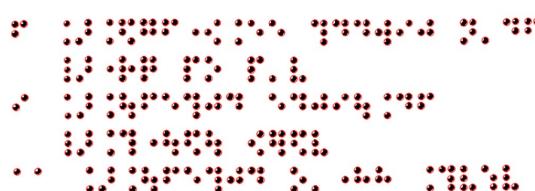
[그림 11] 임계값 처리

3.2 점자 인식 단계

이미지 처리 과정이 끝나면 정제된 이미지를 얻는다. 이제 이 이미지에서 얼룩을 검출하고, 좌표를 만들어서 점자를 분류하여 인식하는 작업을 진행한다.

3.2.1 얼룩 검출

얼룩 검출을 위해 OpenCV 의 simpleBlobDetector class 를 사용한다. 해당 class 는 이미지로부터 얼룩을 검출하기 위해 threshold, findContours 를 사용한다. 해당 클래스를 사용하여 얼룩 중앙의 좌표 값, 얼룩의 크기를 구할 수 있다. 얼룩을 찾기 위한 필터를 설정할 수 있는데, 본 논문에서는 color, convexity, size 를 사용한다. 그림 12 는 simpleBlobDetector 를 통해 이미지의 얼룩을 검출한 모습이다. 모든 점자가 정상적으로 인식된 모습이다.



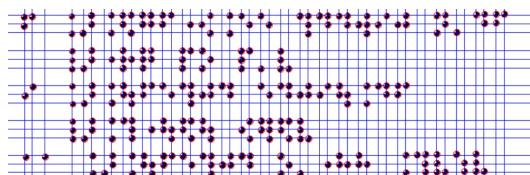
[그림 12] 얼룩 인식 (1)

각각의 점의 좌표를 이용해서 좌표를 만든다. 한 줄에 하나의 선만 긋는 것을 목표로 한다.

맨 첫 줄부터 순회를 시작하여 x 좌표와 y 좌표를 긋는데, 이미 그어진 x 축과 y 축의 좌표와 비교하며, 얼룩 크기 평균값의 절반을 초과하여 떨어져 있지 않다면 같은 줄에 있는 것으로 판단하여 축을 추가하지 않는다.

$$\begin{aligned} coordinateX.push & x_i \text{ if } x_i > x_{i-1} + \frac{\text{blob size avg}}{2} \\ coordinateY.push & y_i \text{ if } y_i > y_{i-1} + \frac{\text{blob size avg}}{2} \end{aligned}$$

축을 그은 이후, 기존에 구한 얼룩들의 좌표를 축에 맞게 조정한다. 이 과정 이후에는 같은 축 위의 점들은 같은 x 값 혹은 y 값을 갖는다. 그림 13 은 해당 방법을 사용하여 축을 그린 그림이다.



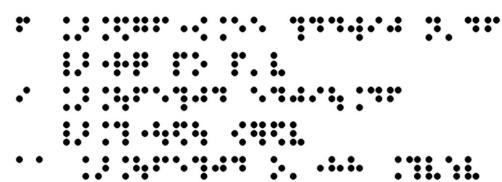
[그림 13] 얼룩 인식 (2)

3.2.2 점자 악보 재 점역

3.2.1 에서 갱신한 좌표를 바탕으로 새롭게 점자를 다시 그린다. 기존의 점자가 전부 모양이 다르게 생겼고, 원이 아니기에 정확한 분류 및 인식을 위해 새롭게 그려준다. 원의 반지름은 얼룩 크기 평균값의 절반으로 설정한다.

$$circle(src, coordinate, blob size avg / 2, color, thickness, line type, shift)$$

그림 14 는 새롭게 그린 점자 악보이다. 이제 이 악보를 토대로 좌표를 갱신하고, 분류 작업을 진행한다.

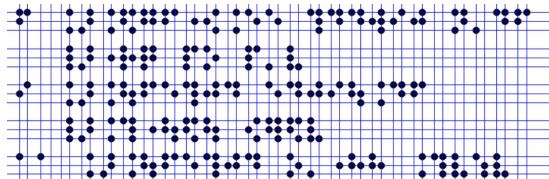


[그림 14] 새롭게 만든 점자 악보

3.2.3 좌표 계산

검출된 점자를 모두 인식했으니, 이제 2x3 개의 점자를 하나의 점자 문자로 묶는 분류 작업이 필요하다. 하지만 그전에, 한 가지 문제점이 존재한다. 지금까지의 방법으로 점자를 검출한다면, 점자가 비어있는 라인이 존재해서는 안 된다. 하지만 그림 14 의 좌측 부분을 보면 공백이 존재함을 알 수 있다. 지금 상태로 분류 작업을 진행한다면, 공백 부분이 인식이 제대로 되지 않아 정상적인 결과를 얻을 수 없다. 이를 축 간격의 평균값을 여러 번 구하는 방식으로 해결한다. 평균값을 계속 구할수록 이상 값들을 계속해서 줄일 수 있고, 정확한 간격을 얻을 수 있다.

정확한 간격을 얻은 뒤, 그려진 축을 순회하며 평균값보다 큰 부분을 메꿔 준다. 그림 15 는 해당 과정 이후에 빈 부분이 모두 채워진 모습이다.

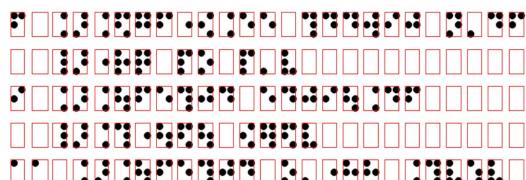


[그림 15] 좌표

3.2.4 분류

만들어진 좌표를 토대로 분류 작업을 진행한다. 2x3 의 크기로 분류 작업을 진행하면 된다. 다음과 같은 방식으로 경계선을 만든다.

```
rect = Rect(Point(X[j] - blob size avg / 2, Y[i] - blob size avg / 2), Point(X[j] + blob size avg / 2, Y[i + 2] + blob size avg / 2))
```



[그림 16] 분류 (1)

그림 16 은 분류 작업을 완료한 모습이다. 모든 점자들이 성공적으로 2x3 크기의 사각형 안에 위치하는 것을 확인할 수 있다.



[그림 17] 점자 순서

추출된 점자를 데이터로 변환하기 위해 2x3 점자에서, 점자가 있는 부분을 1, 없는 부분을 0 으로 간주하여 변환한다. 그림 17 은 해당 과정을 나타낸다.

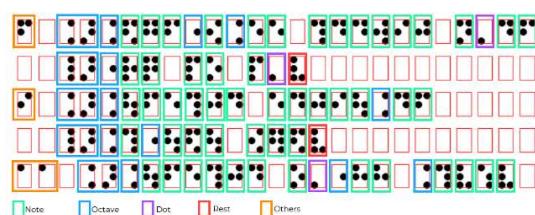


[그림 18] 분류 (2)

그림 18 은 그림 16 과 그림 17 을 바탕으로, 추출한 점자를 모두 10 진수 데이터화 한 모습이다.

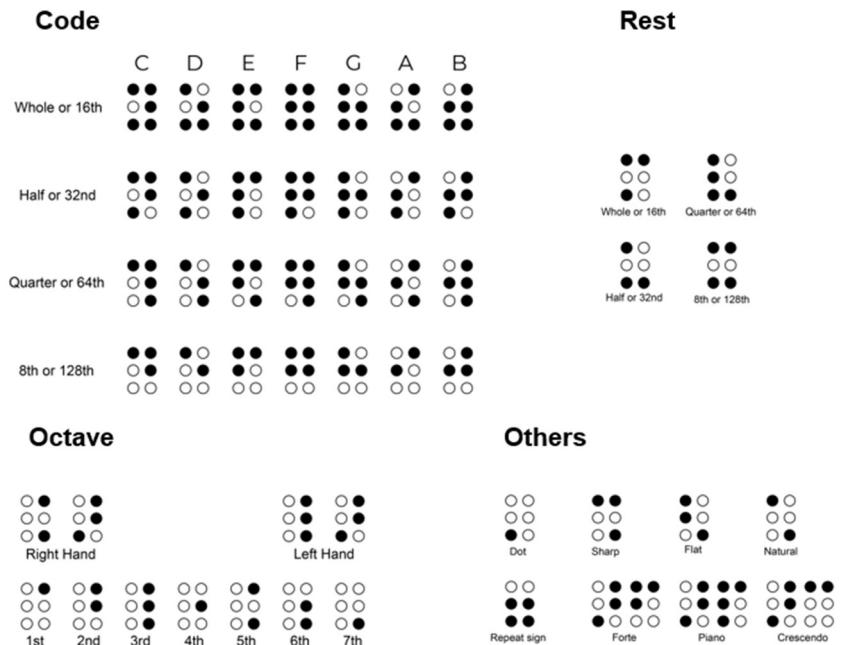
3.2.5 MIDI meta data 생성

이제 추출한 정보를 MIDI meta data 로 변환해야 한다.



[그림 19] 변환

그림 19 는 그림 20 에 정의된 점자 악보 표준에 따라 각 점자 문자 별로 역할을 표시한 것이다. 초록색은 음표 관련 정보이고, 하늘색은 옥타브 관련 정보이고 빨간색은 쉼표 정보이고, 보라색은 점 기호이다.



[그림 20] 점자 악보 표준

| | C | D | E | F | G | A | B | Rest |
|---|------------------|------------|--------------|-------------|----------------|---------------|----|------|
| Whole or 16th | 55 | 39 | 59 | 63 | 47 | 27 | 31 | 50 |
| Half of 32nd | 54 | 38 | 58 | 62 | 46 | 26 | 30 | 35 |
| Quarter or 64th | 53 | 37 | 57 | 61 | 45 | 25 | 29 | 43 |
| 8th or 128th | 52 | 36 | 56 | 60 | 44 | 24 | 28 | 51 |
| Right Hand | Left Hand | Dot | Sharp | Flat | Natural | Repeat | | |
| 21 22 | 17 22 | 2 | 49 | 41 | 53 | 15 | | |

[표 1] 번역 표

| | C | D | E | F | G | A | B | Rest |
|-------------------|----|----|----|----|----|----|----|------|
| 1st Octave | 12 | 14 | 16 | 17 | 19 | 21 | 23 | -1 |
| 2nd Octave | 24 | 26 | 28 | 29 | 31 | 33 | 35 | -1 |
| 3rd Octave | 36 | 38 | 40 | 41 | 43 | 45 | 47 | -1 |
| 4th Octave | 48 | 50 | 52 | 53 | 55 | 57 | 59 | -1 |
| 5th Octave | 60 | 62 | 64 | 65 | 67 | 69 | 71 | -1 |
| 6th Octave | 72 | 74 | 76 | 77 | 79 | 81 | 83 | -1 |
| 7th Octave | 84 | 86 | 88 | 89 | 91 | 93 | 95 | -1 |

[표 2] MIDI meta data

그림 20 은 Louis Braille 에 의해 만들어진 점자 악보의 표준이다. 이 외에도 굉장히 많은 기호들이 존재하지만, 모두 기술하는 것은 생략하겠다.

표 1 은 추출된 데이터가 어떤 의미인지 해석하기 위한 표이다. 표의 내용은 점자 악보를 데이터화 한 십진수 값이고, 그리고 그 값이 어떤 의미를 뜻하는지 알려준다.

점자 악보의 표현 방식은 다음과 같다. 먼저 LH 또는 RH 기호가 나온다. 이는 오른손으로 치는 부분인지, 왼손으로 치는 부분인지 알려준다. 그리고 옥타브 기호가 나온다. 몇 번째 옥타브인지 알려준 뒤 코드 기호가 나온다. 쉼표 기호가 나올 수도 있다. 옥타브 기호에 맞추어서 해당 코드를 연주하면 된다. 옥타브가 변한다면, 다시 옥타브 기호가 등장하여 옥타브가 변했음을 알린다. 도돌이표나 샹, 플랫 등과 같은 기호도 나올 수 있다. 점 기호는 반 박자를 늘리고 싶은 코드 혹은 쉼표 뒤에 등장한다.

이제 해석된 점자 악보를 MIDI 파일로 변환하기 위해서 < TICK / CODE VALUE > 형식으로 변환한다. 박자(tick)의 경우 1 박자를 16 틱으로 계산한다. 온음표의 경우 64 틱이고, 2 분 음표의 경우 32 틱이고, 4 분 음표의 경우 16 틱이고 8 분 음표의 경우 8 틱이고 16 분 음표의 경우 4 틱이다. 점 기호가 등장한다면 해당 박자의 반 박자만큼 더 더해준다. 표 2 는 CODE VALUE 로 변환할 때 사용한다. 표 안의 내용은 MIDI 데이터의 code value이며 각각 어느 음표에 대응되는지 나타낸다. 이러한 정보를 바탕으로 점자 악보 데이터를 변환한다.

| | | | | | | | | |
|-------|-------|-------|-------|-------|------|-------|-------|------|
| 16/64 | 8/65 | 8/64 | 16/57 | 8/62 | 8/62 | 16/60 | 8/60 | 8/60 |
| 16/59 | 8/57 | 8/59 | 48/60 | 8/60 | 8/64 | 16/67 | 8/64 | 8/62 |
| 16/60 | 8/59 | 8/60 | 8/62 | 8/60 | 8/59 | 8/57 | 16/55 | 8/60 |
| 8/64 | 16/67 | 8/64 | 8/62 | 16/60 | 8/59 | 8/60 | 48/62 | 8/55 |
| 8/55 | 16/60 | 16/-1 | 16/62 | 16/-1 | | | | |

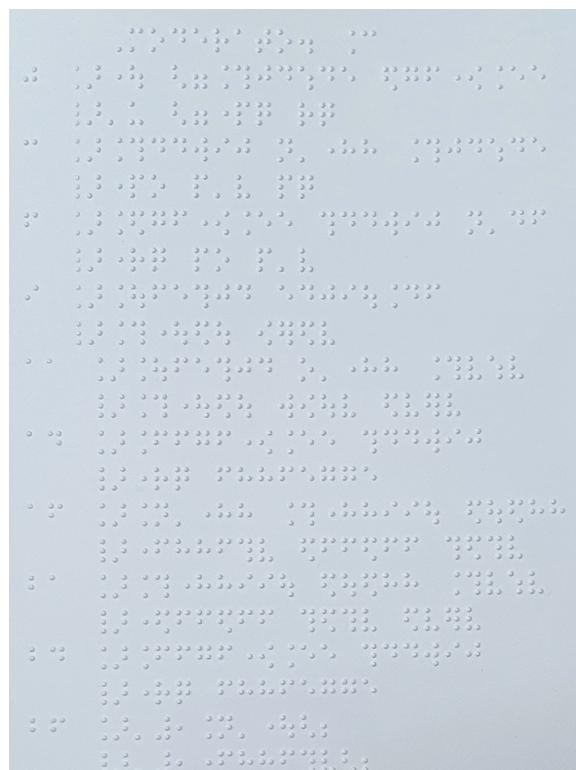
[표 3] 결과

표 3은 표1과 표 2를 활용하여 점자 악보를 MIDI meta data 로 변환한 모습이다. 이제 이 데이터를 MIDI 파일을 만드는 API 에 넣어 재생 가능한 파일을 제작할 수 있다.

IV. 실험 및 토론

4.1 실험 환경

제안된 방법을 사용하여, 실제 점자 악보를 MIDI 파일로 변환하는 실험을 진행한다. “할아버지 낡은 시계” 피아노 점자 악보를 대상으로 실험을 진행한다.



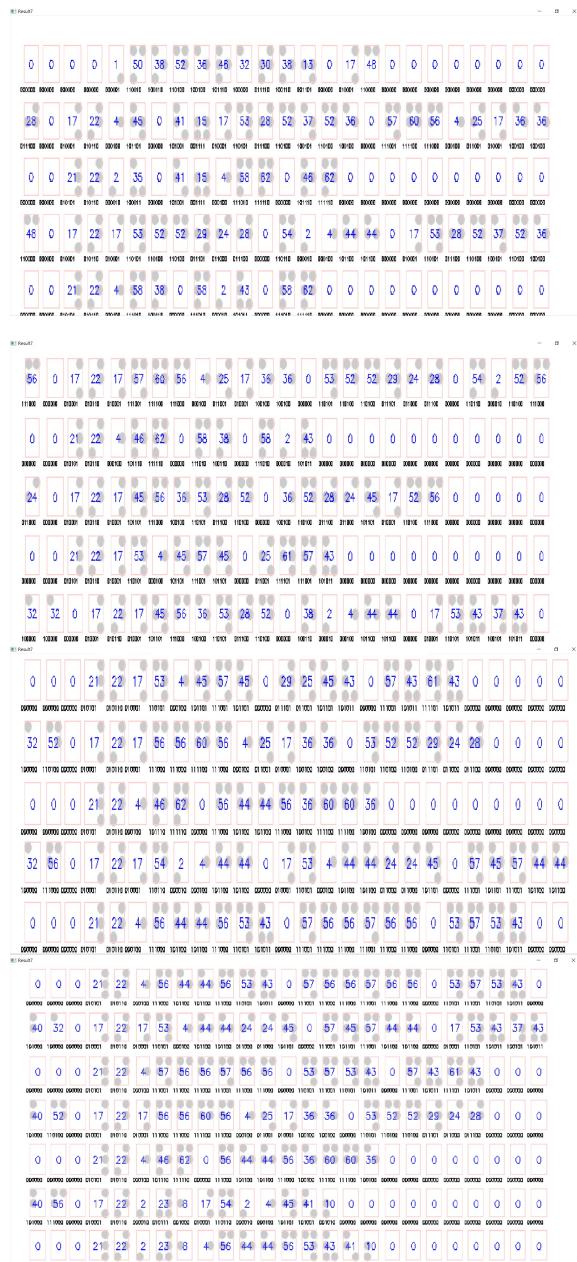
[그림 20] 점자 악보

그림 20 은 실험에 사용한 “할아버지 낡은 시계” 피아노 점자 악보이다. 이 사진을 한번에 인식한다면, 정확성이 매우 떨어지기 때문에 4 분활하여 실험을 진행한다. 4 분활한 사진을 각각 구축된 모델에 input 으로 넣어 MIDI meta data 를 얻고, 얻은 데이터를 수합하여 MIDI 파일을 생성한다.

4.2 실험 결과

4.2.1 데이터 추출

분할한 4 장의 사진을 이미지 처리 과정과 점자 인식 과정을 걸쳐, 10 진수 형태의 데이터로 변환한다. 그림 21 은 4 장의 점자 악보 사진으로부터 데이터를 성공적으로 추출한 그림이다.



[그림 21] 변환된 점자 악보

입력한 모든 점자 악보가 정상적으로 인식되었다. 사진을 분할해서 입력했기 때문에, 추출된 데이터는 4 등분 되어 나온다. 이제, 얻은 데이터를 순서에 잘 맞추어 해석하는 작업을 진행한다.

4.2.2 MIDI 변환

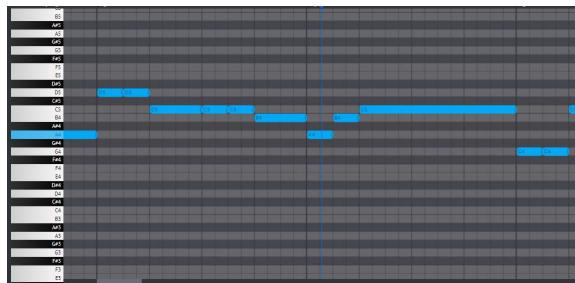
4.2.1 에서 추출한 데이터를 MIDI 파일로 변환할 수 있게, 점자 악보 표준을 따라서 해석한다. 제안 방법에서 제시했던 방식을 사용하여 박자/음정의 값으로 변환한다. 이후 변환된 데이터를 MIDI 생성 API 에 입력하여 MIDI 파일을 생성한다.

| | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|
| 16/55 | 16/60 | 8/59 | 8/60 | 16/62 | 8/60 | 8/62 |
| 16/64 | 8/65 | 8/64 | 16/57 | 8/62 | 8/62 | 16/60 |
| 8/60 | 16/59 | 8/57 | 8/59 | 48/60 | 8/55 | 8/55 |
| 8/59 | 8/60 | 16/62 | 8/60 | 8/62 | 16/64 | 8/65 |
| 16/57 | 8/62 | 8/62 | 16/60 | 8/60 | 16/59 | 8/57 |
| 8/59 | 48/60 | 8/60 | 8/64 | 16/67 | 8/64 | 8/62 |
| 8/59 | 8/60 | 8/62 | 8/60 | 8/59 | 8/57 | 16/55 |
| 8/64 | 16/67 | 8/64 | 8/62 | 16/60 | 8/59 | 8/60 |
| 8/55 | 8/55 | 16/60 | 16/-1 | 16/62 | 16/-1 | 8/64 |
| 8/65 | 8/64 | 16/57 | 8/62 | 8/62 | 16/60 | 8/60 |
| 16/59 | 8/57 | 8/59 | 48/60 | 8/55 | 16/60 | 8/55 |
| 8/55 | 8/57 | 8/57 | 16/55 | 16/52 | 16/55 | 16/52 |
| 8/55 | 8/55 | 16/60 | 8/55 | 8/57 | 8/57 | 16/55 |
| 16/52 | 16/55 | 16/52 | 8/55 | 8/55 | 16/60 | 16/-1 |
| 16/62 | 16/-1 | 8/64 | 8/64 | 8/65 | 8/64 | 16/57 |
| 8/62 | 16/60 | 8/60 | 8/60 | 16/59 | 8/57 | 8/59 |
| 8/62 | 16/55 | 16/55 | 8/57 | 8/57 | 8/59 | 48/60 |
| 16/55 | | | | | | |

[표 4] 최종 결과

표 4 의 값들은 추출된 데이터가 MIDI meta data 로 변환된 값이다. 총 122 개의 음표 정보를 얻었으며, 각각의 음표는 박자와 음정 데이터를 갖고 있다. 이 데이터를 MIDI 생성 API 에 넣고, 재생을 하면 그림 22 와 같이

정상적으로 악보가 재생되는 것을 확인할 수 있다.



[그림 22] MIDI 재생

4.3 토론

기존의 방식은 일반 악보를 점자 악보로 변환하여 점자 악보를 확보하는 것에 치중되어 있지만, 본 논문에서는 이와 반대로 점자 악보를 재생 가능한 일반 악보 형태로 변환하여 이에 대한 접근성을 높이고자 하였다.

본 논문 방식에는 2 가지 한계점이 존재한다. 첫 번째는 건반 악기 외의 악기를 지원할 수 없다는 점이다. 점자 악보의 형식은 성악곡, 민요 같은 음악의 장르에 따라 달라지거나 악기에 따라 달라진다. 본 논문에서는 건반 악기 코드를 기반으로 진행했기 때문에 다른 악기에 대해서는 고려하지 않았다. 이는 추후 연구에서 다른 악기들의 표준도 고려하고, 관련된 표준 규격 데이터도 추가함으로써 해결할 수 있다고 판단된다.

두 번째 한계점으로는, 사진이 기울어졌을 때 인식의 정확도가 급감한다는 점이다. 좌표를 만들어서 점자를 분류하고 인식하는 방식 상 점자들이 한 줄에 있지 않으면 인식에 어려움이 발생한다. 이는 추후 연구에서 인공 신경망을 활용하여 해결할 수 있을 것으로 판단된다. 학습을 통하여 모델의 정확성을 높일 수 있고, 노이즈가 심한 사진이나, 기울어짐이 심한 사진도, 잘 인식할 수 있다고 예상된다.

한계점이 명백히 존재하지만, 모두 추후 연구를 통해 해결 가능한 것으로 판단된다. 본 논문은 점자 악보에 관한 새로운 접근 방법을 제시했다는 점에서 의의를 가진다고 볼 수 있다.

V. 결 론

본 논문은 기존에 제시된 일반 악보를 점자 악보로 변환하는 방식의 역방향인 점자 악보 이미지를 MIDI 파일로 변환하는 방법을 제안하였다. 기존의 방식은 보통의 악보를 점자 악보로 변환하는 방식이었지만, 본 논문에서는 점자 악보를 MIDI 파일로 변환하는 새로운 관점을 제시하였다.

본 논문에서 제시한 Image Processing, Braille Recognition, MIDI Conversion의 3가지 방법을 통해 점자 악보 이미지를 인식하고 분석하여 MIDI 파일을 제작할 수 있다는 사실을 확인했다. 점자 악보 이미지를 MIDI 파일로 변환함으로써 점자 악보로 음악 교육을 원하는 시각 장애인에게 도움을 줄 수 있을 뿐만 아니라 점자 악보 생태계에 긍정적인 영향을 기대할 수 있다.

본 논문에서 제시한 방법에는 건반 악기 코드를 기반으로 진행했기 때문에 다른 악기에 대해서는 고려하지 않았다는 점과, 사진이 기울어지면 인식의 정확성이 떨어진다는 단점이 존재한다.

그러나 제안된 방법을 통해 점자 악보를 MIDI 파일로 변환하는데 성공하고, 이를 활용할 수 있는 가능성을 확인했다는 점에서 의의를 가진다.

향후 연구 방향으로, 인식의 정확성을 높이고, 다른 악기와 다른 음악 장르에 대한 점자 악보의 표준 규격 데이터들을 추가해서 더욱 다양한 상황에서 사용할 수 있도록 한다.

참고 문헌

[1] J Subur, TA Sardjono, R Mardiyanto, "Braille character recognition using find contour method", 2015 International Conference on Electrical Engineering and Informatics (ICEEI) 699-703.

[2] 정지영, 이미애. "전자점자악보의 현황에 입각한 DB구축의 필요성 연구." 디지털융복합연구 15.7 (2017): 343-350

[3] D.Goto, T.Gotoh, R.Minamikawa-Tachino, and N. Tamura, "A Transcription System from MusicXML Format to Braille Music Notation" Hindawi Publishing Corporation EURASIP Journal on Advances in Signal Processing Volume 2007, Article ID 42498, 9 pages

[4] Estevão S. Gedraite, Murielle Hadad, "Investigation on the effect of a Gaussian Blur in image filtering and segmentation", Proceedings ELMAR-2011

[5] 서진범, 장호혁, 조영복, "효율적 딥 러닝을 위한 이미지 전처리 알고리즘 분석", 한국정보통신학회 2020 년도 춘계종합학술대회 논문집 제 24 권 제 1 호 2020.07

[6] Lawrence R. Smith, Chairman, "Music Braille Code 2015", The Braille Authority of North America

[7] 국립중앙도서관 국립장애인도서관 자료 개발과, "개정 점자 악보 제작 지침", 국립장애인도서관.

[8] Braille, https://www.korean.go.kr/front/page/pageView.do?page_id=P000302&mn_id=205

[9] Grayscale, https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

[10] GaussianBlur, https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html

[11] Erosion&Dilation, https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html

[12] Opening, https://docs.opencv.org/3.4/d3/dbc/tutorial_opening_closing_hats.html

[13] Threshold, https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

저자 소개



손 용 락 (Yonglak Son)
승실대학교 소프트웨어 학부에 3학년으로 재학중이며, 수상 경력으로는 2017 융합 캡스톤 디자인 경진대회 최우수상, 2017 빅데이터 경진대회 최우수상이 있다. 관심분야는 머신 러닝 및 컴퓨터 비전과 자연어처리이다.



권 혁 진 (Hyeokjin Kwon)
승실대학교 소프트웨어 학부에 3학년으로 재학중이며 관심분야는 알고리즘 문제해결과 Node.js 프레임워크 기반의 백엔드 기술이다.



주 성 진 (Seongjin Joo)
승실대학교 소프트웨어 학부에 3학년으로 재학중이며, 관심분야는 클라우드 컴퓨팅 기술 Aws와 백엔드 기술인 Spring이다.



박 세 준 (Sejun Park)
승실대학교 소프트웨어 학부에 3학년으로 재학중이며 관심분야는 안드로이드 및 뷰, 리액트와 같은 프론트 엔드 기술과 소프트웨어 아키텍쳐이다.



김 영 종 (Youngjong Kim)
한글과컴퓨터, 씽크프리 연구원, 시큐어소프트 기술이사, 하우리 대표이사, 오픈소스커뮤니티연구소 소장 등을 역임하였으며, 현재 승실대학교 소프트웨어학부 교수로 재직중이다. 관심분야는 보안 및 클라우드컴퓨팅, 블록체인이며 오픈소스기반의 클라우드 및 블록체인 프로젝트를 수행하고 있다.