The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 21, No. 1, pp.69-77, Feb. 28, 2021. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2021.21.1.69
JIIBC 2021-1-10

영상 처리와 딥러닝을 이용한 악보 코드 변환 프로그램

Conversion Program of Music Score Chord using OpenCV and Deep Learning

문지수*, 김민지**, 임영규***, 공기석****

Ji-su Moon*, Min-ji Kim**, Young-kyu Lim***, Ki-sok Kong****

요 약 본 논문은 사용자가 입력한 PDF 악보를 사용자가 원하는 조(chord)의 MIDI 파일로 제공하는 앱의 개발을 다룬다. 이 앱은 사용자가 PDF 악보 파일과 바꾸고자 하는 조를 입력하면 조 변환을 위해 PDF 파일을 PNG 파일로 변환한다. 이를 영상 처리 알고리즘을 통해 악보의 음계를 인식하여 구분하고, 딥러닝을 통해 악보 음표의 박자를 인식하여 구분한다. 이를 통해 사용자가 원하는 조와 기존 악보의 MIDI 파일을 제공한다. 개발한 영상 처리 알고리즘과 딥러닝은 2, 4, 8, 16분 음표, 2, 4, 8, 16분 쉼표, 잇단 음표, 화음 음표가 인식 가능하다. 실험결과 악보의 음표 인식률 100%, 딥러닝 모델을 통한 박자 인식률은 90% 이상인 것을 확인하였다.

Abstract This paper deals with the development of an application that converts the PDF music score entered by the user into a MIDI file of the chord the user wants. This application converts the PDF file into a PNG file for chord conversion when the user enters the PDF music score file and the chord which the user wants to change. After recognizing the melody of sheet music through image processing algorithm and recognizing the tempo of sheet music notes through deep learning, then the MIDI file of chord for existing sheet music is produced. The OpenCV algorithm and deep learning can recognize minim note, quarter note, eighth note, semi-quaver note, half rest, eighth rest, quarter rest, semi-quaver rest, successive notes and chord notes. The experiment shows that the note recognition rate of the music score was 100% and the tempo recognition rate was 90% or more.

Key Words : Music Score, Chord Conversion, MIDI, OpenCV, Deep Learning

Ⅰ. 서 론

음악 전공자가 아닌 사람들은 조표(샤프, 플랫)가 많은 악보를 연주함에 어려움을 느끼는 경우가 많다. 세션 연주자들은 보컬의 키(key)에 맞추어 하나의 조로 맞추어 연주해야 하는 경우가 많다. 또한 대부분의 악보는 저작권 문제로 MIDI 파일로 제공되지 않고 PDF 파일로 제공되어 음표 수정 및 삭제가 불가능한 경우가 많다.

본 논문에서는 영상처리와 딥러닝을 통해 사용자에게 조 옮김 된 MIDI 파일을 제공하는 프로그램을 다룬다.

*준회원, 한국산업기술대학교 컴퓨터공학부 학부생
**준회원, 한국산업기술대학교 컴퓨터공학부 학부생
***준회원, 한국산업기술대학교 컴퓨터공학부 학부생
****정회원, 한국산업기술대학교 컴퓨터공학부 교수
접수일자 2020년 10월 19일, 수정완료 2021년 1월 3일
게재확정일자 2021년 2월 5일

Received: 19 October, 2020 / Revised: 3 January, 2021 / Accepted: 5 February, 2021
****Corresponding Author: kskong@kpu.ac.kr
Dept. of Computer Engineering, Korea Polytechnic University, Korea.

프로그램의 이름은 'OurChord'라 명한다.

OurChord의 주요 기능인 조 변환을 통해 기존 악보와 조 변환 된 MIDI 파일을 제공받아 악보를 수정할 수 있다. PDF 악보를 업로드하여 악기 연주 시 악보를 확인할 수있다. 악기 연주 시 모든 악기를 하나의 키로 맞춰야 할 경우와 현악기 연주 시 각 현의 음을 정확하게 설정해야 하는데 이를 대비하여 튜닝음을 출력하도록 개발하여 올바르게 튜닝했는지 확인할 수 있다.

OurChord 프로그램은 조 변환을 위해 악보 파일을 PNG 파일로 바꾼 뒤 개발한 영상 처리 알고리즘을 통해 악보에 나오는 음표의 음계, 딥러닝 훈련을 통해 생성한 박자 인식 모델을 통해 음표의 박자를 추출한다. 추출한 음표 정보를 통해 PDF 파일의 악보를 수정 가능한 MIDI 파일로 제공하도록 개발하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 본 논문에 관련 된 유사 서비스 혹은 연관성이 높은 프로그램을 비교 분석하였다. Ⅲ장에서는 조 변환을 위한 악보의 음표 인식에 대한 설계 및 구현 내용을 기술하였다. Ⅳ장에서는 악보의 음표 인식 정확도 실험 및 결과를 분석하였다. Ⅴ장에서는 연구 결과 및 향후 계획에 관해 설명하였다.

Ⅱ. 관련 연구

1. 관련 유사 서비스 및 프로그램

다음 표 1에 관련된 제품과 서비스를 보여주고 있다. 이들은 모두 악보 PDF 파일을 제공하지만 forScore와 악보 바다는 유료로 구매해야 사용할 수 있고, forScore의 경우는 악보 PDF 파일을 확인할 수 있지만 수정 가능한 MIDI 파일은 제공하지 않았다. Musecore는 악보 PDF 파일을 MIDI 파일로 변환하여 제공하였지만, 음표 인식 정확도가 매우 낮았다. 악보 바다는 MIDI 파일을 제공하지 않고, 조 변환 악보 PDF 파일을 추가로 구매해야 했다. 조 변환 된 악보 PDF 파일을 얻는데 까지 약 4주에서 8주정도 많은 시간이 소요되는 것을 확인하였다.

표 1. 관련 유사 서비스 및 프로그램
Table 1. Related Service and Program

서비스, 프로그램명	강점	약점
forScore[1]	-악보 음표 기호 스탬프 통해 수정 -MIDI 악기 연결 시 악기 연주 녹음 가능	-비싼 가격 (19,000원) -사용자가 이해하기 어려운 UI
Musecore[2]	-MIDI 파일 제공 -오픈 소스 프로그램	-MIDI 파일 변환 위해 웹 페이지에 접속해야하는 불편함 -음표 인식 정확도 매우 낮음
악보 바다[3]	-다양한 악기의 악보 제공 -다양한 조(chord)의 악보 제공	-비싼 가격 (기본 악보 1,000원 /조 변환 시 5,500원) -조 변환 악보 제공 시 오랜 시간 소요

2. 관련 알고리즘

OurChord는 PDF 악보를 PNG로 변환하여 음표를 인식하기 위해 OpenCV[4]라이브러리와 모폴로지(Morphology) 연산[5]의 침식(Erosion)[6]과 팽창(Dilation)[6]을 사용한다. OpenCV는 실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리로 실시간 이미지 프로세싱에 중점을 둔 라이브러리이며, 모폴로지 연산은 이미지의 밝은 영역이나 어두운 영역을 축소, 확대하는 연산이다.

OpenCV의 템플릿 매칭(Template Matching)[7] 함수는 어떤 이미지에서 특정 템플릿 매칭할 이미지를 검색하고 찾는 방법이다. 템플릿 매칭할 이미지를 타겟 이미지 위에 두고 덮인 이미지의 픽셀 값과 템플릿 이미지의 픽셀 값을 여러 수학 제곱차, 제곱차 정규화, 상관관계, 상관관계 정규화 연산을 통해 비교하여 템플릿 이미지와 일치하는지 전체 픽셀을 확인하여 특정 이미지의 좌표를 얻는다. 연산 방법에 따라 템플릿 매칭 정확도가 다르다. 침식은 필터 영역 내 픽셀 중 최소 픽셀 값을 현재 픽셀 값에 대입하고, 팽창은 픽셀 중 최대 픽셀 값을 현재 픽셀 값에 대입하는 것이다. 팽창을 수행하면 흰 영역이 전보다 넓어지고, 침식을 수행하면 흰 영역이 전보다 좁아지는 형태이다.

CNN(Convolutional Neural Network)[8]은 합성곱 신경망으로 최소한의 전처리를 사용하도록 설계된 다계층 퍼셉트론이다. OurChord는 CNN 을 통해 박자 인식 모델을 구현하여 박자를 확인하였다.

유재명의 논문[9]에서는 음표를 머리, 대, 꼬리 부분으로 분리한 뒤 음표의 머리 부분에 템플릿 매칭을 적용하여 음표를 인식하였다. 하지만 이는 단일 음표만 인식 가능하였다. 본 논문에서는 이를 유돈극의 논문[10]에서 자동차 영상에서 번호판을 추출하기 위해 침식과 팽창을 연속적으로 수행하여 번호판 내의 문자영역을 확장하는 것을 활용하였다. 한영모의 논문[11]에서 템플릿의 방향과 크기를 조정하는 변수를 추가하여 매칭 시 좀 더 안정적인 결과값을 얻을 수 있었던 것을 활용하여 음표 인식 정

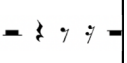
확도를 높였다.

이경민의 논문[12]은 흔들리는 영상을 보정하기 위해 흔들림 영상의 현재 프레임과 이전 프레임 간의 특징점 비교 및 매칭 과정에서 발생되는 특징점 이동을 CNN을 활용하여 이동 방향 및 크기를 추출하여 흔들림 영상을 보정하고, 김정진의 논문[13]에서는 사진을 분류하기 위해 사용자가 본인의 컴퓨터에 있는 사진 파일들을 서버에 전송시키면 서버 측에서 받은 파일들을 CNN을 이용하여 분류하였다. 이에 착안하여 본 논문에서는 CNN을 통해 음표의 박자를 인식하였다.

3. 악보 구성 요소와 MIDI

악보는 기호, 문자, 숫자 등을 통해 음악을 기록한 것이다. 악보는 보통 오선지를 이용해 나타내며 작곡자가 연주가에게 연주에 대한 정보를 제공하기 위한 여러 악보 기호들로 이루어져있다. 여러 악보 기호들을 아래 표 2로 나타내었다.

표 2. 악보 기호
Table 2. Music Sheet Form

기호 이름	악보 기호	설명
오선과 음자리표		-오선은 악보를 그리기 위해 가로로 그은 다섯 개의 줄 -음자리표는 오선의 도 기준 위치를 정하는 기호
조표 (올림표와 내림표)		-올림표, 음에 올림표가 붙는 경우 반음 올림 (파, 도, 솔, 레, 라, 미, 시) -내림표, 음에 내림표가 붙는 경우 반음 내림 (시, 미, 라, 레, 솔, 도, 파)
음표		-음의 높이와 길이 나타냄 -2분 음표, 2박자 연주 -4분 음표, 1박자 연주 -8분 음표, 1/2박자 연주 -16분 음표. 1/4박자 연주
쉼표		-음을 내지 않는 곳과 그 길이 -온 쉼표, 4박자 쉼 -2분 쉼표, 2박자 쉼 -4분 쉼표, 1박자 쉼 -8분 쉼표, 1/2박자 쉼 -16분 쉼표, 1/4박자 쉼

조표가 붙으면 그 음은 반음 올리거나 내려서 연주해야 한다. 조표가 많이 붙은 경우 악기를 연주함에 어려움이 많기 때문에 조표를 없애기 위한 조 옮김을 해야 하는 경우가 많다. 이 외에도 세션 연주 시 보컬의 조에 맞추어 악보를 조 옮김 해야 한다. 조 별 중심음과 그에 따른 음계는 올림표와 내림표의 개수에 따라 달라진다. 아래의 표 3은 올림표와 내림표 수에 따른 조, 표 4는 조 별 중심음에 따른 음계를 정리하였다. 조 옮김은 조표의 개수에 따라 조가 결정되면, 조의 중심음에 따른 음계를 기존 조에서 변환하고자 하는 조의 중심음에 따른 음계로 옮기는 것을 뜻한다.

표 3. 올림표와 내림표에 따른 조
Table 3. Chord by sharp and flat

조(Chord)	올림표, 내림표
C	올림표, 내림표 없음
G	올림표 (파)
D	올림표 (파, 도)
A	올림표 (파, 도, 솔)
E	올림표 (파, 도, 솔, 레)
B	올림표 (파, 도, 솔, 레, 라)
F#	올림표 (파, 도, 솔, 레, 라, 미)
C#	올림표 (파, 도, 솔, 레, 라, 미, 시)
Cb	내림표 (시, 미, 라, 레, 솔, 도, 파)
Gb	내림표 (시, 미, 라, 레, 솔, 도)
Db	내림표 (시, 미, 라, 레)
Ab	내림표 (시, 미, 라)
Eb	내림표 (시, 미)
F	내림표 (시)

표 4. 조 별 으뜸음과 음계
Table 4. Tonality and Melody by Chord

조(Chord)	으뜸음	음계
C	도	도 레 미 파 솔 라 시
C#(Db)	도#	도# 레# 파 파# 솔# 라# 도
D	레	레 미 파 솔 라 시 도
D#(Eb)	레#	레# 파 솔 솔# 라# 도 레
E	미	미 파 솔 라 시 도 레
F	파	파 솔 라 시 도 레 미
F#(Gb)	파#	파# 솔# 라# 도 도# 레# 파
G	솔	솔 라 시 도 레 미 파
G#(Ab)	솔#	솔# 라# 도 도# 레# 파 파#
A	라	라 시 도 레 미 파 솔
A#(Bb)	라#	라# 도 도# 레# 파 파# 솔#
B	시	시 도 레 미 파 솔 라

아래 그림 1은 조 옮김 규칙에 의해 조 옮김 전 기존 A 코드의 악보이고, 그림 2는 B 코드로 변환 된 악보이다. 변환 된 B 조의 악보가 기존 A 조 악보보다 2음 올라간 것을 확인할 수 있다.

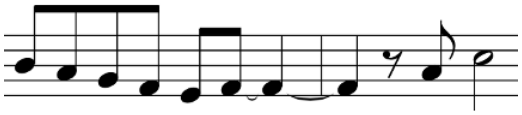


그림 1. 기존 A 조 악보
Fig. 1. Existing A Chord Music Sheet

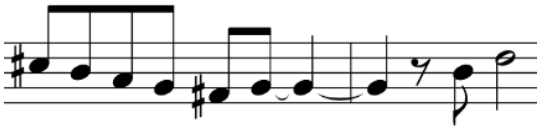


그림 2. 변환 된 B 조 악보
Fig. 2. Converted B Chord Music Sheet

MIDI[14]는 전자 악기 끼리 디지털 신호를 주고 받기 위해 각 신호를 규칙화한 일종의 규약이다. MP3 음악 파일이 연주를 녹음한 것이면 MIDI 파일은 연주를 위한 악보 데이터로 음악을 연주하기 위해 '피아노로 도를 치고 드럼으로 몇 박자로 쳐라'와 같은 명령어가 적혀있는 파일이다. MIDI는 크게 채널과 트랙으로 구성되어있다. 채널은 악기를 의미하고, 트랙은 녹음파일이나, 악보의 구성을 뜻하며, 트랙 하나에 하나의 악기를 통해 생성할 수도 있고, 여러 악기를 섞어 MIDI 파일을 생성할 수 있다. 그림 3은 MIDI 파일 뷰어인 GarageBand[15]를 통해 확인한 MIDI 파일이다.

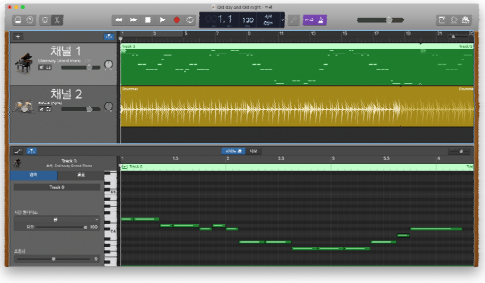


그림 3. MIDI 파일 예시
Fig. 3. Example of MIDI File

Ⅲ. 시스템 설계 및 구현

1. 개발 환경

소프트웨어 개발환경은 다음 표 5와 같다.

표 5. 소프트웨어 개발환경
Table 5. Software Development Environment

종류	언어	개발환경
Server	Python3.7.0	AWS EC2 t2.xlarge
OpenCV	Python3.7.0	Visual Studio Code 1.47
Deep Learning (CNN)	Python3.7.0 Tensorflow[16] Keras[17]	-Intel Core i7-9 9700K -Samsung 970 EVO Plus M.2 2280 (500GB) -G.SKILL DDR4 16G -MSI G-Force RTX 2070 SUPER Gaming X Trio D6 8GB TRI-FROZR
MIDI	Python3.7.0 pyknon[18]	Visual Studio Code 1.47
Application	Java	Android Studio 3.6 Android 9.0 (API 28) NDK 19
DBMS	MySQL	MySQLWorkbench 8.0.18 CE RDS t2.micro

2. 소프트웨어 구성

OurChord 시스템 구성도는 아래 그림 4, 사용자가 악보 파일을 입력하고, 조 변환 시스템 수행 시나리오는 아래 그림 5, 6 과 같다.

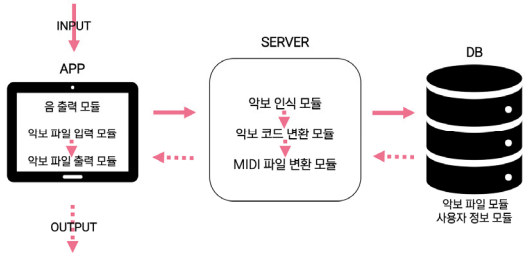


그림 4. 시스템 구성도
Fig. 4. System Configuration Diagram

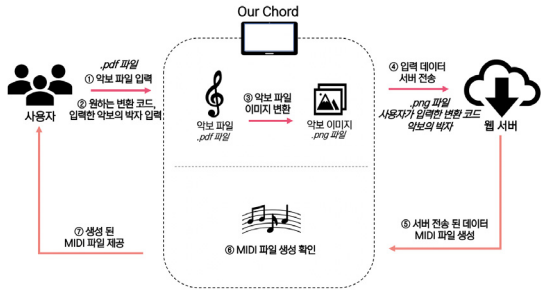


그림 5. 악보 파일 입력 시스템 수행 시나리오
Fig. 5. Scenario of Music Sheet Input System

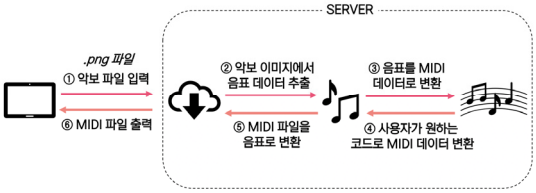


그림 6. 조 변환 시스템 수행 시나리오
Fig. 6. Scenario of Chord Conversion System

위의 그림 5와 같이 사용자가 PDF 악보 파일을 OurChord에 업로드하면 영상 처리를 위해 PNG 파일로 변환하여 서버에 전송한다. 이 후 그림 6과 같이 조 변환을 위해 기존 악보의 코드와 사용자가 변환하고자 하는 코드를 서버에 전송하면 개발한 영상 처리 알고리즘과 딥러닝을 통해 얻은 음표의 음계와 박자 정보를 통해 MIDI 파일을 생성하여 사용자에게 제공한다.

3. 각 모듈 설계 및 구현

가. 음표 음계 인식 모듈

사용자가 업로드한 PDF 악보를 이미지 처리하기 위해 PNG로 변환한다. Ourchord는 탬플릿 매칭으로 음표의 좌표를 인식한다. 템플릿 매칭의 정확도를 높이기 위해 제곱차, 제곱차 정규화, 상관관계, 상관관계 정규화 연산을 차례대로 실험해 본 결과 상관계수 연산 시 템플릿 매칭의 정확도가 가장 높았다. 따라서 템플릿 매칭을 위해 아래 수식 (1)에서 비교할 2개의 이미지를 미리 밝게 보정하여 매칭하고 유사하면 큰 양수, 유사하지 않으면 0에 가까운 음수가 나오는 상관계수 연산을 통해 음표의 좌표를 추출하였다.

$$R(x,y)=\sum_{x',y'}(T(x',y')\cdot I(x+x',y+y')) \qquad (1)$$

템플릿 매칭 이미지와 사용자가 업로드 한 악보에서의 음표 이미지의 크기가 비슷해야 음표 인식 정확도가 높아진다. 따라서 Ourchord는 정확도를 올리기 위해 악보의 음표 크기를 템플릿 매칭할 이미지의 크기와 최대한 맞추기 위해 오선 간격으로 음표의 크기를 알아내어 악보 크기를 조절한다. 악보의 음계를 추출하기 위해 템플릿 매칭을 통한 음계 인식 모듈의 순서도는 아래 그림 7과 같다.

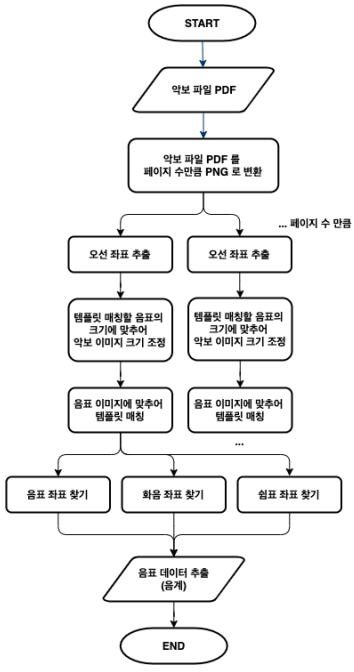


그림 7. 템플릿 매칭을 통한 음계 인식 순서도
Fig. 7. Flowchart of Melody Recognition by Template Matching

템플릿 매칭의 정확도를 높이기 위해 음표 이미지와 인식할 악보의 음표 크기를 확대 및 축소하여 비율을 맞춘다. 이후 템플릿 매칭을 통해 음표 좌표 데이터를 얻고, 이를 오선의 좌표와 함께 활용하여 3옥타브 도부터 6옥타브 도 사이의 음계 데이터를 얻는다. 위의 음계 인식 모듈만으로는 표 2의 음표만 인식 가능하다.

아래 그림 8(이하 “잇단 음표”라 한다.), 그림 9(이하 “화음 음표”라 한다.)와 같은 음표는 음계 인식 모듈에서 추가적으로 필터링하여 음표의 음계를 구분한다.

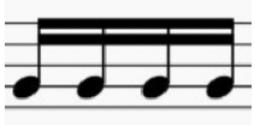


그림 8. 꼬리가 이어진 음표 예시
Fig. 8. Example of Successive Note

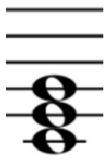


그림 9. 화음 음표 예시
Fig. 9. Example of Harmony Note

잇단 음표를 구분하기 위해 오선 주위의 흰색 배경을 팽창을 통해 흰색 배경을 증폭하여 오선을 지우고, 음표의 머리 부분을 축소시킨다. 이 과정을 거치면 잇단음표의 꼬리부분만 남게 된다. 잇단 음표 외의 꼬리 부분도 남아있을 수 있기 때문에 8분 음표와 16분 음표를 이어주는 꼬리 부분의 연속되는 부분을 찾아 연속 된 순서쌍이 하나의 직선을 이루면 8분 음표, 2개의 직선을 이루면 16분 음표로 구분하였다. 화음 음표를 구분하기 위해 템플릿 매칭으로 음표의 x좌표와 y좌표를 구한다. 음표의 y좌표가 오선의 간격 이상 차이가 날 경우 화음 음표라 간주하고, y좌표에 따라 화음 음표의 각 음계를 구분한다. 악보의 쉼표를 인식하기 위해 템플릿 매칭을 사용하였다. 표 2의 쉼표는 음계가 없고 박자만 존재하기 때문에 템플릿 매칭만을 사용하여 쉼표를 인식하였다. 2, 4, 8, 16분 쉼표, 온쉼표 이미지를 각각 30개씩 총 150개 수집하였다.

템플릿 매칭을 통해 음계를 구분하고, 박자를 인식하기 위해 하나의 음표만 나오도록 이미지를 잘라 악보의 음표 순서대로 저장하였다. 박자 구분을 위한 음표 이미지는 조 변환이 완료되면 삭제되도록 구현하였다.

나. 음표 박자 인식 모듈

음표의 박자를 인식하기 위한 모듈의 순서도는 다음 그림 10과 같다. CNN을 통해 생성된 박자 인식 모델을 사용하여 음표의 박자를 구분한다. 음표 음계 인식 모듈을 통해 얻은 음표 이미지를 순서대로 리스트에 저장하여 각 음표의 박자 정보를 추가하여 MIDI 데이터를 생성한다.

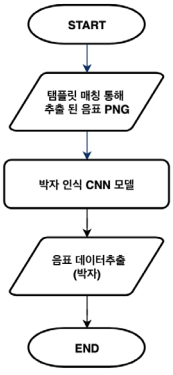


그림 10. CNN 통한 박자 인식 순서도
Fig. 10. Flowchart of Tempo Recognition by CNN

박자 인식 모델을 생성하기 위해 표 6과 같이 음표를 수집하였다. 2분 음표와 4분 음표는 8분 음표와 16분 음표와 비교하여 큰 특징이 없기 때문에 훈련 데이터 수를 줄여 박자 인식 오차율을 줄였다.

CNN을 이용한 박자 인식 모델은 표 7과 같이 3개의 Conv2D, MaxPooling2D, Flatten, 2개의 Dense 레이어를 거쳐 음표의 강한 특징점을 추출하여 음표의 박자를 구분하도록 훈련하여 모델을 생성하였다. 2분 음표와 4분 음표의 배치 사이즈(Batch Size)는 40, 8분 음표와 16분 음표의 배치 사이즈는 20으로 설정하였고, Epoch는 200으로 설정하여 모델을 생성하였다.

표 6. 박자 인식 모델 훈련 및 검증 음표 데이터 수
Table 6. Number of Training and Verification Note Data for Tempo Recognition Model

음표 종류	훈련 데이터 수	검증 데이터 수
2분음표(꼬리 위)	120	60
2분음표(꼬리 아래)	120	60
4분음표(꼬리 위)	120	60
4분음표(꼬리 아래)	120	60
8분음표(꼬리 위)	240	120
8분음표(꼬리 아래)	240	120
16분음표(꼬리 위)	240	120
16분음표(꼬리 아래)	240	120
총합	1440	720

표 7. 박자 인식 모델 층 구조
Table 7. Model Level Structure about Tempo Recognition

Level	Layer	Activation
입력층	Conv2D(224,224,3)	
Level1	Conv2D(222,222,32)	relu
	Conv2D(220,220,64)	relu
	Maxpooling2D(110,110,64)	
	Flatten(77400)	
	Dense(128)	relu
출력층	Dense(4)	softmax

다. MIDI 파일 변환 모듈

음표의 음계와 박자 정보를 통해 MIDI 파일을 생성하는 모듈의 순서도는 아래 그림 11과 같다. 음표 정보를 MIDI 데이터로 바꾸기 위해 위의 표 2의 음표와 같은 한 음만 갖는 리스트와 잇단 음표 정보를 갖는 리스트, 화음 음표를 갖는 리스트, 쉼표 정보를 갖는 리스트를 따로 생성하였다. 이 후 리스트들을 추출한 오선의 순서와 x좌표

를 활용하여 악보의 순서대로 정렬하여 조 변환을 위한 새로운 리스트에 저장하였다.

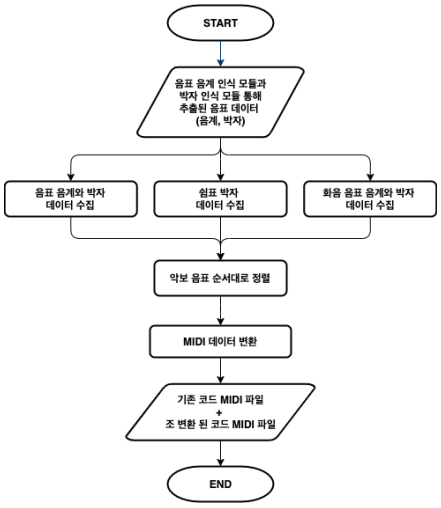


그림 11. MIDI 파일 변환 순서도
Fig. 11. Flowchart of MIDI File Conversion

MIDI 파일 변환 위해 Python 의 pyknon 라이브러리를 사용하였다. pyknon은 음표의 정보와 악보 정보들을 통해 MIDI 파일을 생성하는 라이브러리이다. note, rest, noteseq, midi 함수를 활용한다. note 함수는 음표 하나의 음계와 박자 정보를 입력받고, rest 함수를 통해 쉼표 하나의 박자 정보를 입력받는다. noteseq 함수를 통해 화음 음표의 정보를 입력받는다. 추출한 음표 정보 리스트를 입력 받아 입력 받은 음표 데이터를 seq_chords 함수를 통해 음표의 정보가 모인 새로운 리스트를 생성하였다. MIDI 파일을 생성하기 위해서는 박자 정보가 필요하다. 따라서 생성 된 리스트와 사용자가 입력한 박자를 토대로 MIDI 파일을 생성하였다,

라. 기타 모듈

기타 모듈에는 로그인 및 로그아웃, 개인정보 수정, 악보 파일 저장, 튜닝음 확인 기능이 있다.

사용자가 악보 파일을 업로드 할 때 서버의 데이터베이스에 저장할지 앱 내부의 저장소에 저장할지 선택할 수 있도록 하였다. 사용자가 서버의 데이터베이스에 저장하는 것을 선택한 경우 데이터베이스에 사용자의 악보 PDF 파일을 별도로 분류하는 테이블을 새롭게 생성하고, 사용자가 조 변환을 통해 MIDI 파일을 제공받으면 사용자 별 MIDI 파일을 저장하는 별도의 테이블을 생성하도록 구현하였다.

Ⅳ. 실험 및 결과

악보의 음표 음계를 인식하기 위해 개발한 영상처리 알고리즘과 박자를 인식하기 위한 모델의 정확도를 확인하기 위해 잇단 음표와 화음 음표가 모두 존재하는 악보 10개를 수집하여 악보 인식 정확도를 테스트 하였다. 음표 음계 인식 정확도와 박자 인식 정확도를 아래 표 8로 나타내었다.

표 8. 음표 음계와 박자 인식 정확도
Table 8. Accuracy of Melody and Tempo Recognition

No.	음표 수	음계 정확히 뽑힌 수	박자 정확히 뽑힌 수	음계 정확도	박자 정확도
악보1	89	89	70	100%	78%
악보2	79	79	70	100%	88%
악보3	35	35	35	100%	100%
악보4	35	35	35	100%	100%
악보5	50	50	50	100%	100%
악보6	402	402	399	100%	99%
악보7	227	227	201	100%	88%
악보8	344	344	321	100%	93%
악보9	351	351	302	100%	86%
악보10	511	511	498	100%	97%
평균 정확도				100%	92%

위의 악보 10개에 대한 실험 결과 음표의 음계를 인식하는 영상처리 알고리즘에 대한 정확도는 100% 로 매우 높은 것을 확인하였다. 그에 비해 박자 인식 정확도는 음계 정확도에 비해 낮은 것을 확인할 수 있다.

박자 인식 정확도가 떨어지는 것은 Keras Evaluate 함수[19]를 통해 확인한 박자 인식 모델의 정확도가 낮고, 영상처리 된 이미지를 사용하기 때문이라 판단된다. 따라서 박자 인식 모델 생성 시 훈련 데이터에 영상처리 된 음표 이미지를 추가하고, CNN 레이어, 배치 사이즈, Epoch를 조절하면 모델의 정확도를 더 높일 수 있을 것이라 판단된다.

Ⅴ. 결 론

기존에 개발되어 있는 악보 프로그램 및 서비스는 음표 인식률이 매우 낮고, MIDI 파일을 제공하지 않아 악보를 수정할 수 없었다. 또한 매우 비싼 가격에 비해 음표 인식률이 매우 낮았고, 앱의 접근성이 좋지 않은 경우

가 많았다.

본 논문의 악보 조 변환 프로그램은 기존 프로그램 및 서비스의 단점을 보완하여 잇단음표와 화음음표를 포함한 음계 인식 정확도와 박자 인식 정확도를 높였고, 사용자에게 MIDI 파일을 제공한다. 또한 앱의 접근성을 보완하여 사용자가 쉽고 편리하게 악보를 수정하여 활용할 수 있도록 하였다.

박성우의 논문[20]의 각 신경망 가속화 알고리즘을 사용하여 학습이 끝난 CNN에서 중요하지 않은 가중치를 제거하는 가지치기를 통해 박자 인식 정확도를 높일 수 있을 것이라 판단된다.

References

[1] forScore, https://forscore.co

[2] Musecore, https://musecore.org/en

[3] Akbobada, https://www.akbobada.com

[4] OpenCV, https://opencv.org

[5] Morphological Transformations, https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html

[6] Erosion and Dilation, https://kr.mathworks.com/help/images/morphological-dilation-and-erosion.html

[7] TemplateMatchig, https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

[8] CNN, https://en.wikipedia.org/wiki/Convolutional_neural_network

[9] Jae-Myung You, "Music Score Recognition Robust to Noise Based on Template Matching", Graduate School of Chonnam National University for a Ph.D. Thesis, 2009, pp. 47-48.

[10] Don-Kuk Yu, Vehicle Plate Extraction using Morphology Operation, Diploma Dissertation(M.S) Dept. of Computer Engineering Graduate School, Dongyang University 2002, pp. 5-7, pp. 17-22.

[11] Young-Mo Han, "Two-dimension Automatic Transformation Template Matching for Image Recognition", Journal of the Korea Academia-Industrial cooperation Society, Vol 20, No 9, pp. 1-6, 2019.
DOI : https://doi.org/10.5762/KAIS.2019.20.9.1

[12] Kyung-Min Lee, Chi-Ho Lin, "Video Stabilization Algorithm of Shaking image using Deep Learning", The Journal of The Institute of Internet, Broadcasting and Communication, Vol 19, No 1, pp. 145-152, Feb. 28, 2019.
DOI : https://doi.org/10.7236/JIIBC.2019.19.1.145

[13] Jung-Jin Kim, Sung-Wook Jo, Young-Min Ji, "Image Classification Using CNN", Korea Information Technology Association, Korea Digital Content Association summer joint academic conference thesis collection in 2017.

[14] MIDI, https://ko.wikipedia.org/wiki/MIDI

[15] Garageband, https://www.apple.com/kr/ios/garageband/

[16] Tensorflow, https://www.tensorflow.org/?hl=ko

[17] Keras, https://keras.io/ko/

[18] Geeks & Nerds, "Music for Geeks & Nerds Learn more about music with Python and a little bit of math", Ver. 1.5, pp. 31-41, Mar. 2015.

[19] Keras Evaluate, https://keras.io/ko/models/model/

[20] Sung-Woo Park, Kyong-Ho Han, Woo-Young Jang, "CNN Deep Learning Acceleration Algorithm for Mobile System", Joural of KIIT, Vol. 16, No. 10, pp. 1-9, Oct. 31. 2018.
DOI : http://dx.dio.org/10.14801/jkiit.2018.16.10.1

저 자 소 개

문 지 수(준회원)

• Undergraduate Student at Korea Polytechnic University. Her research interests include Deep Learning and Big Data.

김 민 지(준회원)

• Undergraduate Student at Korea Polytechnic University. Her research interests include Database and Cloud Computing.

임 영 규(준회원)

• Undergraduate Student at Korea Polytechnic University. His research interests include Image Processing and Cloud Computing.

공 기 석(정회원)

• Ki-sok Kong received his BS and MS from Seoul National University in 1984 and 1986, respectively. He received his PhD from KAIST in 1999. He worked at Samsung Electronics, TriGem Computer(Solvit Inc.) and ETRI. He is currently a professor at the department of Computer Engineering at Korea Polytechnic University. His research interests include Operating System, Embedded System and IoT.