

# 타브 코드 인식 및 연주를 위한 시스템 설계와 구현

백병현, 이현종, 오형석, 함종현 황두성  
단국대학교 컴퓨터과학과  
e-mail : byunghyun8@naver.com

## System Design and Implementation for Recognizing and Playing Guitar Tab Chords

Byunghyun Baek, Hyunjong Lee, Hyungseok oh, Jonghyun ham, Doosung Hwang  
Dept. of Computer Science, Dankook University

### 요약

기타를 처음 접할 때 겪는 어려움은 크게 운지법과 곡에 대한 이해다. 연주자마다 곡의 연주 방법이 상이하기 때문에 한 곡에 대해서도 다양한 타브 악보가 존재한다. 이 논문은 타브 악보를 인식하여 연주를 수행하는 휴대 가능한 시스템 설계를 제안한다. 기 연구된 악보 인식은 5선 기반의 악보를 대상으로 하였으며 수평 히스토그램을 사용하여 5선을 제거한 뒤 나타나는 기호들을 인식했다. 본 논문에서는 6선인 타브 악보 인식 및 연주 시스템을 휴대 기기에서도 사용 가능하게 설계하여 많은 연산 양이 요구되는 선 제거 과정을 거치지 않는다. 템플릿 매칭 기법으로 전체 악보에서 타브 악보의 영역을 탐색하고, 탐지된 영역 안에서 선의 시작점을 탐색한다. 선의 시작부터 끝까지 가상 블록을 사용하여 선에 존재하는 공백을 탐지하고, 공백의 문장을 이용해 프렛을 분할하며, 프렛 인식은 프로토타입 기반 분류기를 이용하여 97.0%의 인식률을 보였다.

### 1. 서론

기타를 처음 접하는 이들에게 어려운 점은 운지법과 박자 등이 있다. 또한 처음 보는 곡들에 대해서 음 구성과 박자의 이해가 필요하다. 하지만 연주하기 위해 선정한 곡은 연주자들마다 다른 방식으로 표현하는 경우가 많기 때문에 동일한 곡에 대해서도 타브(tab) 악보가 달라지는 결과가 발생한다. 그 결과로 보유한 악보와 매칭 되는 연주 또는 운지법이 달라짐으로써 기타를 학습하는데 어려움이 발생한다.

악기 연주 학습을 돋는 다양한 앱(app)과 소프트웨어가 개발되었다. 악기의 학습과 관련된 App은 기타, 드럼, 피아노의 UI를 보여주며 터치와 동시에 해당음을 들려준다[1]. Windows 운영체제에서 실행 가능한 Guitar Pro 소프트웨어가 있다[2]. Guitar Pro는 기타 악기를 다루는 많은 사람들이 사용하는 소프트웨어로써 기타 및 베이스 등의 타브 악보 편집 및 제작이 가능하고 인쇄나 시뮬레이션 기능을 갖춘 다 기능 프로그램이다. 이미 악기 연주와 관련하여 많은 앱 또는 특정 운영체제 아래 실행되는 소프트웨어들이 등장하였지만, 악보만을 이용한 연주 시스템은 찾 아보기 힘들거나, 특정 운영체제가 기반이 되는 컴퓨터에서만 사용이 되는 용량이 크고 많은 연산을 요구하는 소프트웨어들만이 존재하기 때문에 시간과 장소에 제한된다.

숫자(digit) 인식은 이미 일상생활에서도 널리 사용되고 있다. 자동차의 번호판을 인식하여 효율적인 차량관리 시스템을 운영하고 있으며[3], Uber에서는 스마트 폰 카메라를 사용하여 신용카드의 숫자를 인식

하는 편리한 결제 시스템을 제공한다.

기존에 연구된 악보 인식 대상은 5선 악보를 대상으로 하며, 악보 인식 과정은 수평 히스토그램을 사용하여 5선을 제거한 뒤 악보에서 사용된 기호들을 인식했다[4,5,6]. 본 연구에서는 6선인 타브 악보에 존재하는 선 제거 과정을 거치지 않고 가상 블록을 이용하여 타브 악보에 인쇄된 프렛 번호를 인식하고 연주를 수행한다.

본 논문에서 설계하는 기타 연주 시스템은 기타 타브 악보의 연주를 수행하는 시스템이다. 2절에서는 기타 연주를 수행하는 시스템의 설계를 제안하고, 타브 악보로부터 프렛 번호를 추출하는 단계를 설명한다. 3절에서는 프렛 번호의 학습기를 제안하고, 4절에서 프렛 번호 인식 평가를 기술한다. 마지막으로 5절에서는 설계된 시스템의 문제점과 기대효과를 토의 한다.

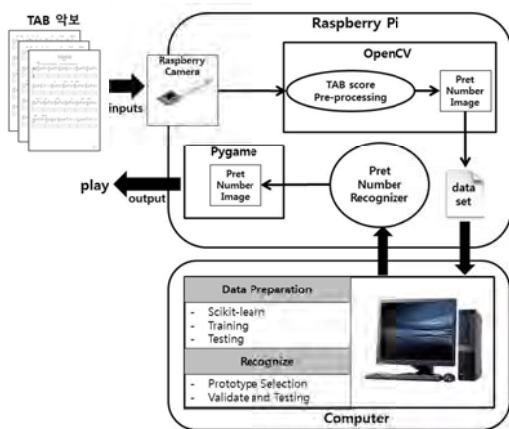
### 2. 제안 방법

#### 2.1. 시스템 개요

개발되는 시스템은 휴대용 기기에서 수행되는 것을 목적으로 한다. 휴대용 기기를 선별하는 기준은 가격 경쟁력과 설계 시스템이 사용될 수 있어야 한다. 기준에 적합한 휴대용 기기로 가격이 저렴한 Raspberry Pi를 기반으로 한다.

그림 1은 타브 악보를 인식하고 연주를 수행하는 과정이다. 타브 악보의 영상 처리를 위한 OpenCV 라이브러리[7]와 연주를 구성하는 음원 파일을 재생 시켜주는 Pygame 모듈[8]을 사용하였다. 프렛 번호의 인

식기는 프로토타입 기반 최근접 이웃 알고리즘을 이용한 분류기를 사용한다[11].



(그림 1) 타브 악보 연주 시스템

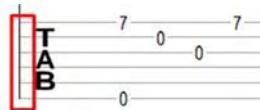
## 2.2. 전처리 과정

그림 2는 타브 악보의 예이다. 타브 악보는 기준의 5선지와는 다른 악보의 표현 방법으로써, 기타의 6현을 그대로 사용하여 6선으로 이루어진 악보이다. 기본적인 구조는 가장 상단의 선이 기타의 1번 줄, 그리고 가장 하단의 선이 기타의 6번 줄을 표현한 것이다. 각 선상의 프렛 번호는 기타 지판에 박혀있는 음의 경계선으로 기타의 음정을 결정하는데 가장 중요한 요소이다. 프렛은 약 0에서 24까지의 번호로 구성되어 있다.



(그림 2) 타브 악보 예

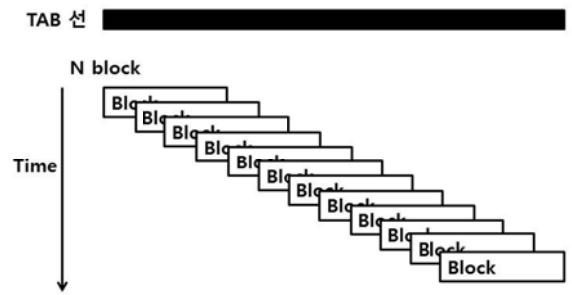
전체 악보에서 기타 연주를 위해 필요한 타브 악보 영역만을 탐색하기 위해서 템플릿 매칭(template matching) 기법을 이용한다. 템플릿 매칭 기법의 정확도를 높이기 위한 목적으로 템플릿의 이진화 작업을 수행한다. 이진화 작업을 수행한 결과로 악보 이미지에서 객체와 배경을 분리하여 정확한 정보를 얻을 수 있다. 이진화를 진행하기 위해 사용되는 임계값은 Otsu 임계값 설정 방법[9, 10]을 이용하여 자동으로 설정한다. 이진화된 타브 악보에서 템플릿 매칭 기법을 사용하여 얻은 좌표값을 기반으로 전체 악보에서 타브 악보의 영역을 추출한다. 그림 3에서 왼쪽 사각형 영역은 모든 타브 악보에서 공통적으로 나타나는 영역이다. 이 영역을 템플릿으로 설정하면 전체 악보에서 타브 악보 영역을 찾을 수 있다.



(그림 3) 타브 악보 영역을 찾기 위한 템플릿

템플릿 매칭의 결과로 반환되는 값은 타브 악보 영역의 좌측 상단 좌표 값들이다. 이 좌표 값을 이용하여 영역을 확대해 타브 악보 영역을 정한다.

6선으로 이루어진 타브 악보 영역에서 세로 1개의 픽셀이 1개의 선이다. 타브 악보 영역의 시작 좌표부터 연속되는 선이 존재하는지 템색함으로써 짧은 연산으로 6개 선의 시작 위치를 찾을 수 있다. 하나의 선에는 여러 개의 프렛 번호가 존재하며, 프렛 번호는 선의 공백에 위치한다. 프렛 번호의 위치를 얻기 위해 가상 블록을 이용하여 공백을 탐색한다. 언급된 가상 블록은 가로 5, 세로 1개의 픽셀의 크기를 가지며 그림 4와 같이 타브 선 왼쪽 시작 부분부터 좌에서 우로 한 픽셀씩 순차적으로 이동할 때마다 생성된다. 블록이 생성되는 범위는 해당 픽셀과 오른쪽으로 인접한 4개의 픽셀을 포함시킨다.



(그림 4) 순차적으로 가상 블록이 생성되는 예

블록을 가로 5픽셀로 설정한 것은 두 가지 이유로 파악할 수 있다. 첫 번째로, 가로 6픽셀 이상의 블록은 분할(segmentation) 시 프렛 번호가 그림 5와 같이 치우치는 문제가 있다. 타브 선에서 하나의 프렛 번호와 다른 하나의 프렛 번호가 6픽셀 이하로 가까운 거리에 위치한 경우에 발생되는 결과다. 한 쪽으로 치우친 프렛 번호 이미지는 프렛 번호가 가지는 특징이 손실되므로 프렛 번호의 인식률이 낮아질 수 있다. 또한, 가로 4픽셀 이하의 블록은 잡음에 비교적 취약하였다. 연속된 픽셀이 공백이 아님에도 공백으로 인식되어 오류가 생길 가능성이 크다.



(그림 5) 분할 오류의 예

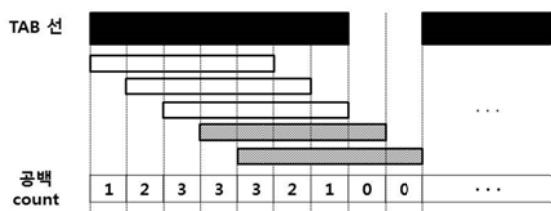
블록을 이용한 공백 탐색 값을 저장하기 위해 그림 6과 같이 타브 선의 가로 픽셀 수만큼 0으로 초기화된 공백 count 배열을 생성한다.

TAB 선	[REDACTED]	[REDACTED]
공백 count	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

(그림 6) 초기화 된 공백 count 배열

타브 선의 각 픽셀은 공백 count 값을 갖는다. 각 픽셀마다 생성되는 블록은 해당 픽셀과 오른쪽에 인접한 4개의 픽셀을 포함하여 총 5개의 픽셀에 대해 공백 유무를 검사한다. 5개의 픽셀 모두 선으로 판단되면 블록에 포함되는 각 픽셀과 매칭 되는 공백 count 값을 1증가 시킨다. 반면, 블록에 포함되는 5개의 픽셀 중 하나 이상의 픽셀이 공백으로 판단되면 블록에 포함된 각 픽셀과 매칭 되는 공백 count 값을 변화시키지 않는다.

그림 7은 블록을 사용하여 공백 count 값을 갱신시키는 간단한 예이다. 각 픽셀마다 생겨나는 블록은 5개의 픽셀을 포함하고, 서로 다른 5개의 블록이 한 픽셀에 대해 공백 유무를 탐지하기 때문에 공백 count 값은 0~5의 범위를 갖는다. 타브 선의 끝에 도달하면 최종적으로 공백 count 값의 배열을 얻는다.



(그림 7) 공백 count 배열을 계산하는 예

계산된 공백 count 배열의 인덱스 0부터 배열의 크기 M까지 순차적으로 공백 count가 3인 값을 탐색한다. 처음 공백 count 값이 3인 N위치와 다음 공백 count 값이 3인 N+1위치까지 공백으로 판단한다. 그림 8은 공백을 탐색하는 예를 보여준다.

TAB 선	[REDACTED]	[REDACTED]
공백 count	5 5 5 4 3 2 1 0 0 1 2 3 4 5 5 5	

(그림 8) 공백 count 배열에서 공백 탐색의 예

타브 선에서 프렛 번호가 존재하는 공백은 가로로 약 6~10픽셀의 크기를 갖는다. 블록의 크기 단위로 공백을 탐지하고, 공백 count 배열에서 3인 값을 선과 공백의 경계 값으로 설정한다. 따라서 1~2개의 잡음인 픽셀이 존재하더라도 정확한 공백을 탐색할 수 있다.

프렛 번호 인식기를 구현하기 위해서 타브 악보에서 나타나는 프렛 번호의 데이터가 필요하다. 프렛 번호를 수집하기 위해 초보자들이 연주할 수 있는 악보를 선별했다. 선별된 악보는 13개의 클래식 곡과 18개의 외국 곡, 그리고 22개의 한국 곡이다. 53개의 악보에서 수집한 약 2만 4천 개의 프렛 번호를 바탕으로 학습 과정을 거쳐 인식기를 구현한다. 그림 9는 타브 선에서 공백을 탐지한 후, 분할 한 결과로 나타나는

프렛 번호의 예이다.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

(그림 9) 프렛 번호의 분할

### 3. 프렛 번호 인식기와 연주

프렛 번호 인식을 위해 선택 기반 최근접 이웃 알고리즘을 사용한다[11]. 주어진 프렛 번호 데이터는 각 프렛 번호 클래스를 대표할 수 있는 프로토타입 집합의 새로운 훈련 데이터이며, 분류 학습은 최근접 이웃 규칙을 이용한다. 훈련 데이터로부터 각 데이터가 중심이 되어 클래스 영역 내 대표하는 구(sphere)을 생성한다. 구의 반지름은 최대 거리의 동일 클래스 데이터와 최소 거리의 다른 클래스 데이터와 중간으로 선택한다.

새로운 프로토타입 데이터의 선택 방법은 집합 덮개 문제(set covering optimisation)로 정의하며, 클래스 별 소수의 프로토타입을 선택하게 된다. 프로토타입은 가능한 많은 동일 클래스 데이터를 대표하는 데이터를 우선으로 선택한다. 새 프렛의 클래스는 가장 가까운 프로토타입의 클래스로 예측한다.

연주를 구성하는 음의 정보와 순서를 저장하기 위해 음원 실행 값을 사용한다. 음원 실행 값은 프렛 번호의 종류, 프렛 번호의 가로 위치와 세로 위치를 포함한다. 약 0~24까지의 범위를 갖는 프렛 번호는 각기 다른 음으로 표현된다. 프렛 번호 인식기를 이용하여 프렛 번호의 종류를 음원 실행 값에 포함시킨다. 연주는 타브 악보 영역에서 좌에서 우로 순차적으로 음을 발생시켜 완성된다. 프렛 번호가 존재하는 가로 위치는 연주를 구성하는 음의 순서를 알기 위해 음원 실행 값에 포함한다. 타브 악보의 각 선마다 음의 차이가 존재한다. 따라서 프렛 번호가 어떤 선에 위치하는지 알기 위해 세로 위치를 음원 실행 값에 저장한다. 타브 악보의 전처리 과정이 모두 끝나면 최종적으로 연주를 구성하는 음의 순서와 정보가 음원 실행 값에 저장된다.

음원 실행 값은 미리 저장되어 있는 음원 파일의 파일명으로 변환한다. Pygame 모듈을 사용하여 변환된 파일명과 매칭 되는 음원 파일을 실행시켜 전체 연주를 구성한다. Pygame 모듈은 동시에 음원 파일을 8개 까지 실행시킬 수 있어 원활한 연주가 가능하다.

### 4. 실험결과

표 1은 수집된 53개의 악보로부터 추출한 프렛 번호별 데이터의 수이다. 프렛 번호별 약 400~4600개의 데이터가 수집되었다. 프렛 번호 16부터 25까지는 수집된 악보에서 나타나지 않았다. 프로토타입 기반 학습 성능 평가는 5 식 교차 검증(5-way cross-validation)으로 수행하였다. 학습 성능을 위하여 최근접 이웃 알고리즘, 베이지안 학습기, 그리고 프로토타입 학습 알고리즘과 비교되었다.

&lt;표 1&gt; 실험에서 사용된 프렛 번호와 개수

프렛	수	프렛	수	프렛	수	프렛	수
0	4,595	4	1,131	8	1,593	12	1,293
1	1,068	5	1,853	9	1,175	13	509
2	2,428	6	837	10	1,414	14	441
3	2,252	7	2,212	11	385	15	506

표 2는 준비된 학습데이터로부터 생성된 테스트 데이터 대한 F-score 평가결과이다. 프로토타입 기반 분류 학습은 16개 프렛 코드들에 대해 평균 약 98.0%로 나타났다. 프렛 0, 5, 6, 9, 10, 15 등의 경우 100.0% 테스트 결과를 보여 분류 학습에 준비된 데이터의 수가 충분하기 때문에 분석된다. 그러나 프렛 3, 8의 경우 학습데이터의 추가가 요구된다. 이러한 결과는 베이지안 학습보다 높으며, 최근접 이웃 규칙(1-KNN)과 유사한 결과이다. 한편, 프로토타입 기반 학습에서 선택된 프로토타입의 비율은 약 10.0~25.0%로 적은 수의 프로토타입으로 최근접 이웃 규칙과 대등한 일반화 성능을 나타냈다. 그러므로 하드웨어 제한을 갖는 Raspberry Pi 기반 시스템의 구현에 프로토타입 기반 학습기의 활용이 더 높다

&lt;표 2&gt; 학습성능 비교

프렛	베이지안	1-KNN	제안방법 (1-KNN)
0	0.72	1.00	1.00
1	0.99	0.99	0.99
2	0.99	0.99	0.99
3	0.99	0.99	0.84
4	.0.99	0.99	0.99
5	0.98	0.99	1.00
6	0.99	1.00	1.00
7	1.00	0.99	0.99
8	0.67	1.00	0.88
9	0.84	1.00	1.00
10	0.99	1.00	1.00
11	1.00	1.00	0.99
12	0.98	0.99	0.99
13	0.79	1.00	0.99
14	0.99	0.99	0.99
15	0.64	0.99	1.00
평균	0.91	1.00	0.98

## 5. 결론

본 연구에서는 타브 악보를 바탕으로 연주를 실행 시켜 곡에 대한 이해를 돋고, 휴대용 기기에서도 실행 가능한 시스템을 제안하여 시간과 장소에 제한 받지 않고 학습의 효율을 높였다. 연산 양을 최소화시키기 위해 기존에 연구된 악보 인식 방법에서 5 선을 제거하는 과정을 배제하고, 가상 블록을 사용하여 음을 찾았다.

타브 악보는 프렛 번호 외에 다양한 기호들이 존

재하지만 본 연구에서 개발된 시스템은 다양한 기호들을 배제하고 프렛 번호만 처리하였다. 또한, 인쇄된 타브 악보에서 나타나는 6선이 수평이라고 가정했다. 마지막으로 프렛은 0~24까지의 범위를 갖기 때문에 프렛 데이터가 충분히 확보된다면 모든 프렛을 인식할 수 있다.

## 참고문헌

- [1] B. K. Hwang, "An Implementation of Smartphone-based Multiple Musical Instruments Application supporting Social Playing," Journal of Digital Contents, Vol. 12, No. 4, pp. 575-583, Dec. 2011.
- [2] Guitar Pro, <http://www.guitar-pro.com>
- [3] S. H. Jung, Y. W. Kwon, C. B. Sim, "An Efficient Car Management System based an Object-Oriented Modeling using Car Number Recognition and Smart Phone," Journal of Korea Institute of Electronic Communication Science, Vol. 7, No.5, pp. 1153-1164, Oct. 2012.
- [4] G. H. Park, S. Y. Oh, H. J. Son, J. M. Yoo, S. H. Kim, G. S. Lee, "Decision-Tree Algorithm for Recognition of Music Score Images Obtained by Mobile Phone Camera," Journal of The Korea Contents Association, Vol. 8, No. 6, pp. 16-25, May. 2008.
- [5] K. B. Kim, W. J. Lee, Y. W. Woo, "Automatic Recognition and Performance of Printed Musical Sheets Using Fuzzy ART," Journal of Korea Institute of Electronic Communication Science, Vol. 6, No. 1, pp. 84-89, Feb. 2011.
- [6] J. M. Yoo, G. H. Kim, G. S. Lee, "Music Recognition by Partial Template Matching," Journal of The Korea Contents Association, Vol. 8, No. 11, pp. 85-93, Nov. 2008.
- [7] OpenCV Document , <http://docs.opencv.org>
- [8] Pygame, <http://www.pygame.org>
- [9] J. R. Parker, "Algorithms for Image Processing and computer vision 2nd Edition," wiley publishing, pp.149-151, 2011.
- [10] Puneet, Naresh Kumar Garg, "Binarization Techniques used for Grey Scale Images," International Journal of Computer Applications, Vol.71, No.1, pp. 8-11, June 2013.
- [11] S. Y. Shim, D. H. Hwang, "Prototype based Classification by Generating Multidimensional Spheres per Class Area," Journal of The Korea Society of Computer and Information, Vol. 20, No. 2, Feb. 2015.