선 추적과 템플릿 매칭을 이용한 악보 인식 시스템

이이삭, 최나영, 김인중
한동대학교 전산전자공학부
kortie00@gmail.com, ainaying@naver.com, ijkim@handong.edu

Music Score Recognition System Using Line Tracking and Template Matching

Isaac Lee, Na-Young Choi, In-Jung Kim
Dept. of Computer Science & Electronic Engineering, Handong Global University

요 약

본 논문에서는 악보 영상을 인식하고 이를 연주할 수 있는 시스템을 제안한다. 이 시스템은 스캐너 또는 카메라로 악보 영상을 입력 받는다. 먼저 입력 영상을 전처리하여 영상 분석에 알맞은 형태로 변환시키고 선 추적으로 오선을 추출한 후, 템플릿 매칭을 이용해 음표, 쉼표, 보표, 조표 등을 추출하여 인식한다. 그리고 인식한 결과를 MIDI로 출력한다. 이 악보 인식 시스템을 통해 여러 실험 데이터를 검토해 본 바, 본 시스템이 실용적임을 보였다.

1. 서론 및 관련연구

음악은 인간의 정서와 감정을 나타내는 예술로써, 다양한 분야에 쓰인다. 음악은 일반적으로 악보에 의해 표현되는데 사람이 이러한 악보로 표현된 음악을 연주하기 위해서는 특별한 훈련이 필요하다.

악보를 자동으로 인식하여 연주가 가능하다면 악기를 배우지 않은 사람이라도 편리하게 음악을 감상할 수 있을 것이다.

본 시스템은 여러 기호가 복합되어 존재하는 이미지 속에서 악보 인식을 통해 음표, 조표 등의 요소만을 추출할 수 있다. 그리고 이를 MIDI로 연주할 수 있다.

악보의 자동 인식에 관한 연구가 1960년대 후반부터 시작되었다. 기존의 악보 인식 방법에서는 오선 인식을 하기 위해 주로 투영법[1]을 이용하였고, 기호를 인식하기 위한 방법으로 통계적 방법[2], 신경망을 이용한 방법[3], 구조적 방법[4] 등이 있다. 전통적인 방법으로써 통계적인 방법은 각 특징을 이용한 입력 패턴과 표준 패턴을 비교하고, 구조적 인식 방법은 음악 기호의 구조적 정보를 제공하여 패턴을 분해해 나간다. 최근에 들어와 신경망을 이용한 방법에 대한 많은 결과가 나오고 있다. 많은 수의 단순한 프로세스들을 연결한 시스템을 사용하여 패턴을 인식한다. 하지만 악보의 특성상 복잡하고 다양하여 현재까지 모든 형태의 악보를 인식할 수 있는 악보 인식 시스템은 존재하지 않는다.

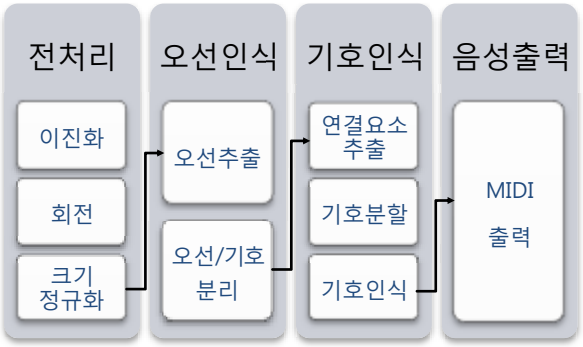
본 논문에서는 악보 영상을 받아 선 추적 알고리즘을 통해 오선을 추출하고, 계층적 구조로 구성된 템플릿 매칭을 사용하여 음표를 인식시킨다. 그리고 추출된 결과를 MIDI로 출력하는 시스템을 제안한다. 2장에서는 시스템 개요에 대해, 3장에서는 전처리, 오선인식, 기호인식, 음성출력과 같은 시스템의 주요 기능

에 대해, 4장에서는 실험 및 결과에 대해 설명하겠다.

2. 개요

본 시스템의 대략적인 흐름은 다음과 같다.

영상을 입력 받아, 이진화한 후, 이미지를 인식에 용이한 형식으로 변환한다. 그 다음, 오선 추출, 연결요소 추출, 기호인식 등의 추출과정을 거쳐 최종적으로 MIDI로 출력한다. 이 시스템의 구성도는 그림 1과 같다.



[그림 1] 시스템 구성도

3. 주요기능

본 시스템의 주요기능은 전처리, 오선 추출 및 제거, 기호인식, 음성 출력 네 단계로 나눌 수 있다.

전처리

전처리는 입력된 악보의 영상을 악보인식과정에 용이한 형태로 변환하는 작업을 하는 단계이다. 이 단계는 이진화, 영상회전, 크기정규화 과정으로 나눌 수 있다.

입력 영상은 일반적으로 스캔을 통하거나 카메라를 통한 영상이기 때문에 기울기, 왜곡의 문제가 있다. 전처리 단계에서는 이 같은 영상을 개선하여 인식하기 용이한 형태로 변환한다. 우선 입력 영상이 컬러영상일 경우 수식 1과 같이 변환을 시켜 각 픽셀의 R, G, B 세 값을 단 하나의 명도영상 값으로 바꾼다.

$$Gray\_Data = 0.299 \times B + 0.587 \times G + 0.114 \times R$$

[수식 1] Gray화

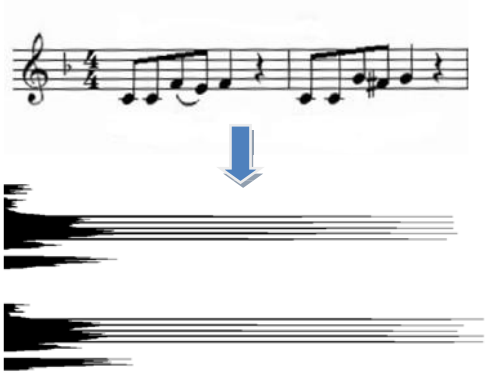
명도영상의 각 픽셀데이터는 0~255사이의 값을 갖는다. 이진화에는 적응 이진화, 평균 이진화, otsu 이진화, 등 많은 방법들이 있지만 악보영상은 보통 극단적으로 흑과 백으로 화소가 나뉘어져 있는 경우가 많기에 본 시스템에서는 단순 임계치를 주어 이진화하는 전역적 이진화[6]를 사용하였다.

이진화가 끝난 후 이미지의 기울임을 보정하기 위해 최상단의 수평선을 추적하여 첫 오선을 추출한 후 기울어진 영상을 반대방향으로 회전한다.

기호인식에서 동일한 마스크를 적용시키기 위해 크기를 정규화한다. 오선의 높이로 이미지 크기를 변경하는데 이는 악보 이미지 상에 오선의 차지하는 비율은 각각 다르지만 오선의 2줄 사이에 위치하는 오선 사이의 거리와 음표 머리크기의 비율은 일정하다. 오선 한 단(5줄)의 높이를 측정하여 오선의 높이가 미리 정의한 값이 되도록 이미지의 크기를 변경한다.

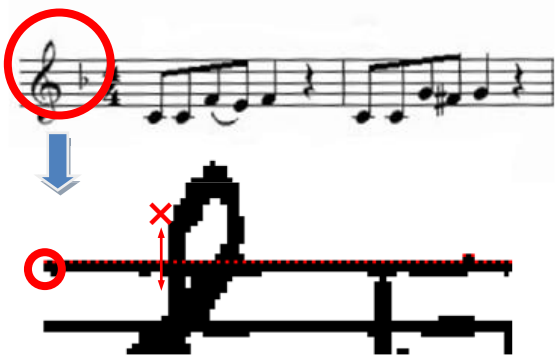
오선인식

오선인식 단계는 오선을 추출한 데이터를 바탕으로 모든 기능이 동작하기 때문에 악보 인식 시스템에서 가장 중요한 부분이다. 이 단계는 오선 추출과 오선/기호 분리의 두 과정으로 나눌 수 있다.



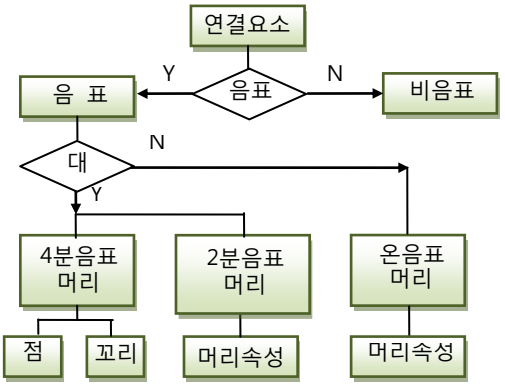
[그림 2] 수평 히스토그램

오선 추출의 알고리즘을 살펴보면 의 기존 연구에서는 그림 2와 같이 수평히스토그램 분석을 통해 오선을 추출하였다. 수평히스토그램 방법은 영상 회전 시 발생하는 잡영에 약한 단점이 있다. 스캔 과정에서 발생한 잡영 또는 기울임 보정 시에 발생한 잡영이 있는 경우, 좀 더 강인한 오선 추출 방법이 필요하다. 본 시스템에서는 그림 3과 같이 오선의 시작점을 찾은 후 수평으로 오선을 추적하는 방법과 수평히스토그램 분석으로 구한 데이터를 OR연산하여 오선의 데이터만 가진 이미지를 생성하였다. 이것은 그림 3과 같은 왜곡된 환경에서 수평히스토그램 방법보다 좀 더 잡영에 민감한 데이터를 구하기 위한 방법으로서 오선의 굵기가 일정하지 않더라도 오선을 추적할 수 있다. 기본적으로 연결요소 추출 알고리즘을 생각할 수 있지만 그림 3의 높은음자리표와 같이 오선과 연결되어 있는 요소들은 굵기가 임계치보다 큰 성분을 검출하여 배제할 수 있다. 단, 이 연결요소 방법으로는 끊어진 오선을 찾을 수 없는 단점이 있기에 수평히스토그램으로 추출한 오선 데이터와 OR연산하여 좀 더 명확한 오선을 추출할 수 있다.



[그림 3] 선 추적(Line Tracking)

오선을 추출하게 되면 본 영상에서 오선을 분리하기 위해 오선을 추출한 이미지와 비교하여 오선을 제거할 수 있다. 하지만 오선과 붙어있는 음표 같은 굳이 오선 부분을 제거할 필요가 없는 기호들은 배제함으로써 원 음표나 쉼표, 보표 등을 그대로 보존한다. 이 오선인식 단계를 거침으로서 오선으로부터 분리된 기호들만 남은 악보를 인식할 수 있다.

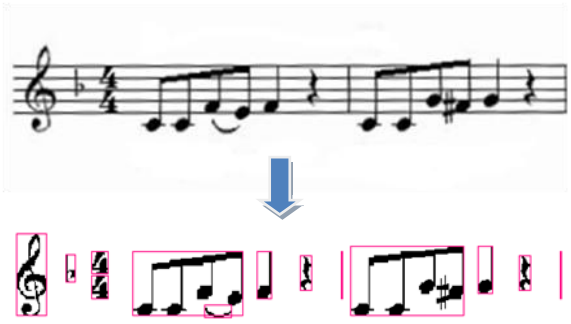


[그림 4] 음표의 계층 구조

기호 인식

기호 인식 단계는 오선성분이 제거된 영상에서 각각의 기호를 인식하는 단계이다. 이 단계는 연결요소 추출, 기호 분할, 기호 인식 과정으로 나눌 수 있다.

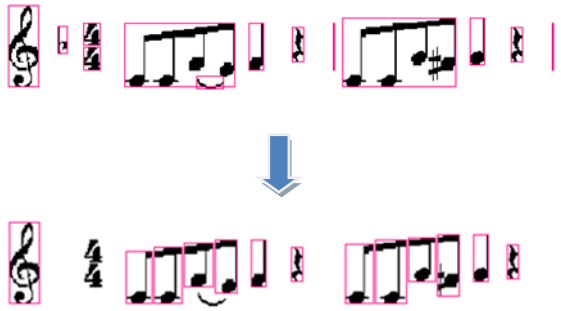
연결요소 추출 과정은 인접한 흑화소들을 하나의 객체로 만드는 작업이다. 그림 5는 연결요소 추출의 결과이다. 그러나 그림 5에서 볼 수 있듯이 여러 음표가 하나의 연결요소로 나타나는 경우가 있으므로 연결요소와 기호가 정확히 일치하지는 않는다.



[그림 5] 연결요소 추출

따라서 이 음표들을 각각의 다른 음표로 인식시키기 위해서 분리시킬 필요가 있다. 즉, 음

표로 인식된 연결요소 속에 여러 개의 음표머리가 검출이 되면 그림 6와 같이 각각의 음표로 나눌 수 있다. 이 과정을 거쳐 각각의 기호들을 온전히 추출할 수 있다.



[그림 6] 음표분리 및 잡영제거

이 과정 후에 예를 들어 가사 같은 악보상의 잡영 또는 인식대상이 아닌 연결요소를 제거하기 위해서 각 연결요소들의 종류를 결정해야 한다. 이 때 악보상의 보표, 조표, 음표, 쉼표 등을 인식하기 위해서 템플릿 매칭 방법을 사용하였다. 그림 7은 본 시스템에서 사용한 템플릿의 예이다. 여기에서 쓰인 템플릿 매칭 방법은 어떤 템플릿이 종류는 다르지만 크기가 큰 연결요소에 잘못 인식되는 점을 막기 위해 가장 큰 템플릿부터 작인 템플릿 순으로 비교하는 순서로 이루어 졌다. 템플릿 매칭 방법을 사용하여 각각의 음표, 쉼표, 조표 등으로 인식된 부분을 제외하고는 악보상의 잡영으로 볼 수 있으므로 쉽게 잡영을 제거할 수 있다.

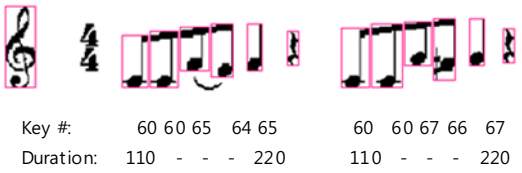
[그림 7] 템플릿의 예

음표의 음정인식 부분은 이 단계 후에 MIDI로 출력하기 위해 필요한 데이터인 음표를 가지고 오선의 데이터와 비교하여 어떤 음을 가졌나 판단할 수 있다. 또한, 이 음표들은 템플릿 매칭으로 음표대, 8분 음표의 꼬리 등의 존재 여부를 알아내어 박자를 정할 수 있다. 그림 5에서 보인 분리된 음표들은 각 꼬리부분의 흑화소들의 높이를 계산하여 음표의 박자를 정할 수 있다.

기호 인식과정을 거치게 되면 악보를 출력시키기 위한 음표, 쉼표 등의 요소들만 남게 된다.

음성 출력

음성출력부분은 전 단계에서 처리된 연결요소의 정보를 음성으로 출력한다. 음성출력은 General MIDI를 이용하여 작성되었다. 음표와 쉼표로 구성된 데이터를 가지고 각 음표의 음정에 해당하는 숫자와 음표와 쉼표의 박자를 MIDI출력의 형식으로 변환하여 음을 출력한다. 즉, 음악을 일련의 숫자로 변환하는 것이다. 이 변환된 숫자들을 이용하여 음성을 출력시키게 된다.



[그림 8] 음표의 음정과 박자

4. 실험

실험에 사용한 시스템은 펜티엄4 3.0GHz, 512M RAM, Windows XP에서 Visual C++ 6.0을 이용한 환경이었다. 실험에 사용한 데이터들은 평평하게 스캔되어 회전 외의 굴곡 같은 왜곡이 없다는 조건을 만족하는 100개의 악보를 사용하였으며 총 3번씩의 실험을 통하였다. 이 악보들 중 1/3은 템플릿의 규격과 같은 악보, 2/3는 새로운 데이터로 초등학교 음악교과

서에 실린 악보를 대상으로 하였다. 실험결과는 표 1과 같이 나왔다.

	템플릿 규격에 맞는 악보	새로운 데이터
인식률	97.8%	89.7%

[표 1] 실험 결과

5. 결론 및 향후 발전방향

논문에서 제안한 방법을 이용한 실험을 통해 적절한 환경에서 성공적인 악보 인식을 수행하였다. 본 시스템에서는 선 추적과 템플릿 매칭 알고리즘을 사용하여 이미지화 된 악보를 인식해 보아 그것이 실용적임을 보았다. 선 추적 방법과 수평 히스토그램을 사용한 오선 추출 방식은 다양한 악보에서 오선을 찾는데 유용하였다. 음표인식 부분에서는 템플릿 매칭 방법을 사용하였는데 이러한 템플릿 매칭 방법에 의한 인식은 매우 강력하고 쉽게 추출할 수 있다는 장점과 함께 입력되는 영상의 정규화 시의 왜곡에 불리한 단점이 있다. 즉, 아주 작은 크기의 악보를 정규화시키면 규격화된 템플릿보다 작은 요소들이 나타나 인식률이 떨어진다. 이 부분은 복수의 템플릿을 사용하는 방법이나 신경망 등 다른 인식방법을 함께 적용시켜 보완해야 할 부분이다.

악보 인식뿐만 아니라 기타 이미지 인식은 여러 외부 요인에 민감하다. 이러한 제약 조건들은 앞으로 계속 연구되어야 할 부분이다. 보다 신뢰성 있는 이미지 인식을 위해 외부 요인에 큰 영향을 받지 않도록 하는 연구가 이어져야 하겠다.

참고문헌

[1] 박철우, 황인성, 정동석, "인쇄 악보의 자동 인식 및 연주를 위한 연구", 대한전자공학회 학술대회논문집, 제16권 2호, p653~656 (1993)

[2] R.GNANADESIKAN, J.R.KETTENRING, "Pattern Recognition and its Applications in Industry", Peoceedings of International Conference on Statistical Methods and Statistical Computing for Q and P, 제1권, p330~336(1995)

[3] 신재욱, 이성기, "신경망을 이용한 악보 인식에 관한 연구", 대한전자공학회 학술대회논문집, 제3권, p451~454(1993)

[4] 이기돈, "악보인식에 관한 연구", 박규태박사 회갑 논문집, p139~151(1994)

[5] 김정규, "음악기호의 계층구조를 이용한 인쇄악보 인식", 한국화상학회지, 제3권 1호, p66~74 (1997)

[6] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, Prentice Hall

[7] Charles Petzold, Programming Windows, Microsoft Press