한국음향학회지 제26권 제7호 pp. 334~342 (2007)

화음 이름과 음계 분석을 이용한 호모포니 4부 합창 악보의 자동 조성 검출 알고리듬

Automatic Tonality Detection Algorithm of Homophony 4-Part Chorus Sheet Music Using Chord Names and Scale Analysis

이 강 성*, 이 돈 웅**
(Gang Seong Lee*, Donoung Lee**)
*광운대학교 교양학부 컴퓨터공학 **서울대학교 음악대학 작곡과
(접수일자: 2007년 8월 2일; 수정일자: 2007년 10월 12일 채택일자: 2007년 10월 24일)

MusicXML 파일로 표현되는 수직적으로 화음을 결정할 수 있는 충분한 정보가 있는 호모포니 (homophony) 4부 합창 형식의 음악에서 화음 이름을 자동으로 판단하고 사용된 음계와 검출된 화음 이름을 이용하여 조성을 자동으로 검출하는 알고리듬을 기술한다. 화음 이름은 사용된 조에 관계없이 분석이 가능한 구성 화음의 절대적인 이름이나 환경에 따라 두 개 이상의 화음 이름으로 결정될 수 있는 여러 상황이 존재하게 되는데, 몇 가지 파라미터를 이용하여 상황에 가장 적절한 화음을 선택하는 알고리듬을 기술한다. 또한 사용된 음들을 이용하여 음계를 추정하고, 구해진 화음 이름과 추정된 음계를 이용하여 음악의 조성을 파악하는 알고리듬을 기술한다. 조성이 결정되었으면 다시 조성과 파악된 조성을 기반으로 화음을 표기하고 MusicXML 파일로 출력한다.

핵심용어: 악보 분석, 음악 분석, 자동 조성 검출, MusicXML

투고분야: 음악음향 및 음향심리 분야 (8)

This paper presents an algorithm for the automatic detection of chord names, scales and tonalities from music file, expressed in MusicXML format which has enough information to determine harmonies vertically like 4-part choir. Chord names are absolute names which can be used and analysed independently of the tonality. An algorithm selecting the best chord name is described, which can decide the most appropriate one from ambiguous situations. Candidate musical scales are extracted using the notes in a given time window. The tonalities of the music are determined using the chord names and candidate scales. The final output format of the process is also MusicXML file with chord names, marked non-harmonic notes, relative harmonic symbols and tonalities.

Key words: Sheet music analysis, MusicXML, Tonality detection

ASK subject classification: Musical Acoustics and Psychoacoustics (8)

I. 서 론

음악 분석이란 상대적으로 단순한 요소들로 음악적 구조를 표현하고 그 구조 안에서 이러한 요소들의 기능을 구명 (究明)하는 것이다. 화성 분석은 이러한 목적을 이루는 가장 근본적인 음악 분석으로 사용된 조성과 화음의 기능을 분석하는데 있다. 이러한 분석 작업은 특별히 훈련된 전문가들에 의해서만 가능한데, 자동 분석이 가능하다면 화성분석과 조성 분석을 통해서 곡의 상위 수준의 구조 (조성, 악절 및 구조등)도 쉽게 파악할 수 있을 뿐만 아니라, 이러한 결과를 누적시킨 데이터베이스를 기반으로 여러 가지 흥미로운 연구와 응용 프로그램을 만들어낼 수 있다. 예를 들면 어떤 작곡가의 패턴을 분석해 그와 유사한 알고리즘 작곡을 가능하게 할 수도 있고, 화성진행이나 조성에 따른 곡의 검색, 유사한 조적 변화 특성을 갖는 곡의 검출 등이 가능하다. 또한 어떤 감정 표현을 위한 화성 진행의 통계적 특성과 같은 분석등도 가능하

책임저자: 이 강 성 (gslee@kw.ac.kr)
139-701 서울시 노원구 월계동 447-1 광운대학교 교양학부 컴퓨터공학전공

다. 본 논문은 자동 화음 결정과 이를 기반으로 한 자동 조성 결정 알고리듬에 관한 연구이다.

높이가 다른 두 개 이상의 음이 동시에 울릴 때 이를 화음 (和音, chord)이라고 하며, 일정한 법칙에 따라 연결된 화음을 화성 (和聲, harmony)이라고 한다 [1]. 화음 이름 (chord name)은 사용된 조성에 관계없이 부여되는 전대적인 이름이다. 화음 이름을 결정하기 위해 수직적 음들을 고려할 때 일정 구간내의 음 구성에 따라 두 개 이상의 화음으로 결정될 수 있는 여러 상황이 존재하게 되는데, 이러한 후보 화음의 목록을 얻어내고 환경과 파라미터에 따라서 최적의 화음을 결정해야 할 필요가 있다.

음악 분석에 있어서 조성의 결정과 화성의 분석은 서로 분리할 수 없는 관계이다. 조성이 결정되어야만 상대적인 혹은 기능적인 화성 표기가 가능하며, 그러기 위해서는 적절한 화음 분석이 선행되어야 조성을 결정할 수 있다. 예를 들어 화음 분석의 결과로 얻어지는 C 코드는 C장조에서는 1도 화음 (I)이지만 F장조에서는 5도 화음이고 (V), G장조에서는 4도 화음 (IV)이다. 따라서 화음 분석을 기준으로 조성이 결정되고 나면 화성 표기를 할 수 있다.

조성 분석을 위하여 Temperley는 조성 음악 이론에 근거한 기본 규칙들을 가지고 화성 분석을 수행했으나 작품 분석을 하는데 그의 기본 규칙을 적용하는데 한계가 있었고 (예:바흐의 프랑스 모음곡 5번) [2], Pardo 와 Birmingham [3]가 개발한 HarmAn이란 시스템은 세그먼트를 구분하고 하나의 코드를 부여하는 방식으로 분석을 했으나 음표 입력을 수치 값 (미디 노트 정보)으로만 활용했기 때문에 신볼을 처리해야 하는 부분에 한계가 있었다. 예를 들어 F#과 Gb을 구분할 수 없었다. 또한 Winograd [4]는 조성분석의 선구자적인 작업을 수행했는데, 체계적인 문법을 이용하여 악보를 분석하는 방법을 발표했다. 그의 방법은 미리 수작업으로 악보를 4부의 완벽한 화음의 열로 변환한 후에 처리하도록 하고 있다. 이 과정에서 장식음이나 경과음과 같은 비 화성음이 모두 제거되어 처리되었다. 그 이외에도 여러 방법들이 개발되었지만 앞서 소개한 방법을 포함하여 대부분 화음 이름의 진행을 만들고 그 진행에 따라 조성을 결정하는 알고리즘을 사용하였다.

화성 분석의 기초가 되는 화음 분석에 있어서 화음의 결정은 다음과 같은 이유들로 인해서 어려움을 겪는다 [6].

1) 장식음과 불완전한 구성음들

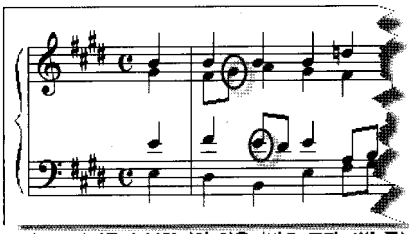

그림 1. 장식음과 불완전한 화음 (바흐 코랄 4번 중)
Fig. 1. Non-harmonic notes and incomplete chord (from Bach Choral No. 4).

화음 혹은 화성 분석 중에 부딪히는 가장 흔한 문제는 비화성음의 존재이다. 이것은 음이 너무 많은 경우로 대부분 전타음, 계류음, 경과음과 같은 장식음에서 온다 [1] (그림 1에서 원 안의 음들). 반대의 경우는 구성음이 적어서 분석이 쉽지 않은 경우이다.

2) 중의성

같은 구성음이라고 할지라도 상황에 따라 여러 의미를 내포하는 경우가 있다. 예를 들어 그림 2의 왼쪽 악보에서 4분 음표 단위로 분석할 경우 네모 안의 화음을 ii (2도 화음-Am)로 보고 F#을 비화성음으로 볼 수 있지만, vii° (F#dim-7도(감)화음, 제1전위)으로 보고 E를 비화성음 혹은 7음으로 해석할 수도 있다. 8분음표 단위로 분석할 경우는 ii, vii° 로도 분석이 가능하다. 그림 2의 오른쪽 악보에서 동그라미 친 부분을 I로 보고 $E^{\flat}$과 G음을 비화성음으로 간주할 수도 있고, IV^{6}_{4} (제2전위)화음으로 볼 수도 있다.

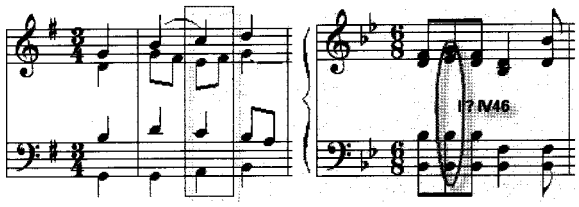

그림 2. 화음의 중의성
Fig. 2. Ambiguity of a chord.

3) 규칙적이지 않은 화성적 리듬

전통적인 화성학에서 화성적 리듬이 규칙적이라고 가정할 수는 없다. 다시 말해서 화음이 박이나 마디 단위로 동일하다는 가정을 할 수 없다. 화성은 어느 시점에서도 변할 수 있다.

4) 화성 규칙의 다양성

'화성학'이란 화성적 법칙을 정하는 것이 아니라 전통적인 화성의 사용 관습을 정리해 놓은 것이므로 모든 경우에 적용되는 화성적 규칙을 설정하기 쉽지 않다.

본 논문에서는 이러한 어려운 점을 극복할 수 있는 화음 분석 알고리즘을 이용하여 화음 분석을 하고, 그 결과

를 이용하여 다시 조성을 분석하도록 하였다. 앞서 다른 기법들이 화음을 분석하고 다시 분석된 화음이 조성 시스템에 얼마나 맞는지를 검토한 반면, 본 논문에서는 사용된 음을 고려하여 음계를 추정하고 그 음계 위에서 적용 가능한 조성들에 중에서 최적 조성을 결정하도록 하였다. 이것은 전체 조를 대상으로 검색하는 것보다 검색 범위를 좁힐 뿐만 아니라 그 정확도도 높일 수 있다.

음계를 추정하고 그 추정된 음계를 대상으로 조성을 결정하는 예를 들면, 음들이 C, D, E, F, G, A, B가 사용되었다고 하면 우선은 C장조일 가능성이 가장 높다. 하지만 A 에올리안 선법일 가능성도 배제할 수 없다 (선법은 여기서의 고려대상이 아니다). 만일 G 대신에 G#이 사용되었다고 한다면 A단조일 가능성이 더 높아진다. 하지만 G음과 G#음이 함께 사용되었다면 가능성은 C장조, A단조, G장조로 늘어난다. 이러한 음에 대한 정확한 정보 표현은 MusicXML [5]을 이용함으로 가능하게 되었다. MIDI에서는 음을 단지 숫자로만 표현하므로 G#과 Ab을 구분할 수 없었는데 반해 MusicXML 파일은 이러한 것을 정확하게 표현하므로 좀 더 상위 수준에서의 심볼 처리가 가능하다.

음계와 함께 중요한 정보는 화음의 진행이다. 예를 들어, 코드 진행이 C->F->G->C 라고 한다면 이것은 A 단조이기 보다는 C 장조일 가능성이 더 높다. G음과 G#음이 함께 사용된 경우라도 C->Caug->F->G 와 같은 진행이 가능하기 때문에 코드의 진행이 조성 내에서 이루어지는 가를 고려해야 한다. G#음이 Am7->Dm->E->Am 상황 하에서 사용된 것이라면 이것은 A 단조일 가능성이 높아진다. 조성음악의 화음은 토닉과 도미넌트를 중심으로 진행되기 때문에 이들의 발생 빈도나 진행 관계 혹은 종지관계를 보면 조성을 결정할 수 있다.

본 논문에서의 정보 처리 절차는 1) 사용된 음을 이용해 후보 음계를 추정하고, 2) 그 음계 후보를 이용해 후보 조성 (장단조)을 구하고, 3) 각 후보 조성에 대해 코드들의 적합도를 계산해 가장 최적인 조성을 선택한다. 그리고 4) 선택된 조성을 기반으로 다시 상대적인 화성 표기를 하는 과정을 거친다. 전체적인 블록도를 그림 3에 보인다. MusicXML 악보는 후보 화음 이름을 구하는 부분과 음계를 추정하는 부분으로 입력된다. 구해진 후보 화음 이름 결과와 후보 음계는 조성 결정 블록에 전달된다. 조성 결정 블록은 음계 후보와 후보 화음을 읽어들여 조성을 결정하고, 조성에 맞는 화성표기를 한다.

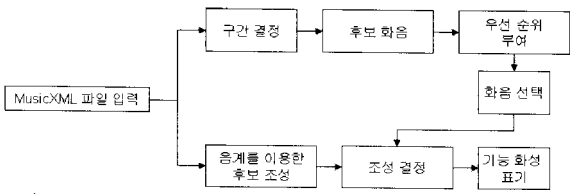

그림 3. 화음 이름 분석과 조성 결정 블록도
Fig. 3. Block diagram of the system.

II. 화음 이름 분석 알고리듬

2.1. 화음 분석 구간 결정

화음 분석을 하기 위해 정해야 할 기준 중에 하나는 얼마의 구간을 기준으로 분석할 것인가 하는 문제이다. 즉, 최소 단위의 분석 음가 기준이 정해지면 최소 단위보다 작은 음가들은 최소 단위 음가로 묶여서 분석된다. 최소 단위 구간은 하나의 화음이 최소한 유지되는 구간으로 아무리 많은 음이 있어도 하나의 화음으로 분석되는 구간이다. 예를 들어 최소 단위가 4분 음표라면, 16분 음표들은 4분 음표 단위로 묶여서 분석된다. 하지만 음이 충분하지 않으면 화음 분석이 가능할 때 까지 분석 창의 크기를 늘려야 한다. 기본 최소 구간을 박으로 설정하고 최대 구간을 마디로 설정했다. 물론 이것은 파라미터로 변경 가능하다. 한 구간의 분석이 끝나면 연속된 시점부터 계속 분석을 해나간다. 모든 분석이 끝난 후 동일한 화음을 갖는 구간들은 통합된다 [7].

2.2. 화음 이름 추출

구간이 주어지면 그 안의 음들은 다음과 같은 방법으로 후보 화음 이름들이 선택된다.

(1) 사용 가능한 모든 코드 이름에 대하여 화성음/비화성음 음들을 구분한다.
(2) 빠진 음의 개수와 전위 정보를 계산한다.
(3) 화음 구성음이 모두 채워진 경우, 화음 후보에 등록한다.
(4) 음이 생략된 경우는 다음 각 경우를 만족하는 경우에만 화음 후보로 등록한다.
   가) 반감화음인 경우는 근음, 3음, 7음이 존재한다.
   나) 장, 단 3화음인 경우, 근음과 3음이 존재한다.
   다) 장, 단 3화음에 단7도 음이 부가된 화음이 경우는, 비화성음이 없는 경우에는 근음과 7음이 존재하거나, 비화성음이 있는 경우에는 근음, 3음, 7음이 존재하는 경우에 후보로 등록한다.

이 단계에서 결과로 얻어지는 후보로 코드 정보는 다음과 같은 필드를 갖는다.

(코드이름, 전위, [화성음의 발생빈도], 생략된 화성음의 수, [상대적 비화성음들])

예를 들면 다음과 같다.

('Dm7', 3, [1, 1, 0, 1], 1, [2])

즉, 어떤 구간에서 나타난 음들을 분석한 결과 코드 이름은 'Dm7'이고, 제3전위 (24화음) 형태를 취하고 있으며, 화음 구성음들 중에 5음은 0회 나타났고 ([1, 1, 0, 1]) 나머지 음들은 각 1회 발생했으며, 따라서 생략된 화음 구성음의 수는 1개이고, 마지막의 [2]로부터 근음 D로부터 2개 반음이 올라간 음 (E)이 비화성음으로 사용되었다는 것을 알 수 있다. 비화성음들은 근음을 기준으로 표시하는 것이 좋기 때문에 근음에 대한 상대적 값으로 변환되었다. 이명동음 (예:G#, Ab)인 경우도 모두 목록에 포함된다. 즉, 이름 없이 MIDI 음높이만 주어진 61, 65, 68, 73 (C#4, F4, G#4, C#5) 경우에는 다음과 같은 두 개의 결과가 나온다.

[('C#', 0, [2, 1, 1], 0, []), ('Db', 0, [2, 1, 1], 0, [])]

하지만 심볼 이름이 주어진 C#4, F4, G#4, C#5인 경우에 대해서는 다음과 같은 결과만 나온다.

[('C#', 0, [2, 1, 1], 0, [])]

이 단계에서 처리 대상으로 삼은 화음의 종류는 다음과 같다.

장3화음 / 단3화음 / 속7화음 / 단3화음+단7음 / 장3화음+장7음 / 단3화음+장7음 /

감화음 / 감7화음 / 반감화음 / 계류화음 (sus4) / 계류화음+단7음 / 증3화음 / 장3화음+6음 / 단3화음+6음

2.3. 후보 코드 이름들의 정렬

앞서의 방법으로 구해진 코드 이름들은 나름대로의 가능성을 갖고 있는데 가능성이 높은 코드 순서대로 정렬할 필요가 있다. 코드 이름 정렬 기준은 다음과 같다. 모두 오름차순 정렬이며, 가장 앞선 것이 가장 가능성이 높은 화음으로 간주된다.

(1) 생략된 화성음의 수,

(2) 비화성음수,

(3) 전위,

(4) 코드이름

코드 이름에 의한 정렬은 가독성을 높이기 위함이며 가능성과는 관계없다. 이들의 선택은 조적 상관관계가 고려되어야 한다. 그림 4에 대한 후보 정렬의 결과를 다음에 보인다.

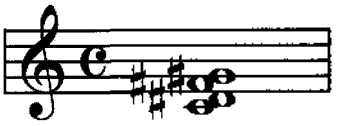

그림 4. 코드 선택 문제
Fig. 4. Chord name selection problem.

[('Ab7', 1, [1, 1, 1, 1], 0, []),
('G#7', 1, [1, 1, 1, 1], 0, []),
('Cdim', 0, [1, 1, 1], 0, [8]),
('Ab', 1, [1, 1, 1], 0, [10]),
('G#', 1, [1, 1, 1], 0, [10]),
('D#m6', 3, [1, 1, 0, 1], 1, [5]),
('Ebm6', 3, [1, 1, 0, 1], 1, [5]),
('D#m', 4, [1, 1, 0], 1, [5, 9]),
('Ebm', 4, [1, 1, 0], 1, [5, 9])]

2.4. 정렬 후 선별

구해진 목록 중에서 후보 가능성이 있는 것만을 취하고 그렇지 않은 것은 버린다. 구해진 코드 이름들 중에 생략된 화성음의 수가 0이고 비화성음이 없는 것이 존재할 경우에는 그렇지 않은 것들은 모두 후보에서 제외된다. 생략된 화성음의 수가 0이고 비화성음이 없는 것이 없다면 앞서 구해진 전체가 후보 코드 이름들이 된다. 예를 들어 앞 단에서의 그림 4의 목록을 처리한 결과는 다음과 같다.

[('Ab7', 1, [1, 1, 1, 1], 0, []), ('G#7', 1, [1, 1, 1, 1], 0, [])]

2.5. 화음군 나누기

앞 단계에서 얻어진 목록은 1단계 후보와 2단계 후보군으로 나누어진다. 1단계 후보군은 비화성음이 없는 화음들로 다음과 같은 경우이다:

(1) 생략된 음이 없는 화음

(2) 5음이 생략된 장단3+7화음

(3) 근음과 7 음이 생략되지 않은 속7화음

(4) 5음 생략 3화음

1단계인 경우 계류화음 (sus4), 장6도 부가화음, 장7도 부가화음이, 2단계인 경우는 장6도 부가화음인 경우가 비화성음 처리를 위하여 후보에서 제외되었다.

2단계 후보군은 1단계를 분류하는 기준과 같으나 비화성음이 존재하는 경우에 분류된다. 만일 1단계 화음들이 있다면 2단계 화음군은 후보 대상에서 제외되며, 1단계

화음들이 없는 경우에는 2단계 화음군이 후보 대상으로 오르게 된다.

2.6. 화음군에서 화음 선택하기

같은 화음군에서의 화음들은 다음과 같은 경우에 제거되고 나머지만 선택된다:

(1) 딴이름 같은 코드 제거하기: 예를 들어 C#dim7, Gdim7은 전위만 다를 뿐 같은 코드이다. 이 경우 전위가 높은 것이 제거된다.

(2) 같은 조건이라면 불안정한 코드를 제거한다: 예를 들어, 그림 5에서와 같이 C, E, G, D음이 있을 경우에 C코드의 add 9으로 볼 수도 있지만, Gsus4 add 6로 볼 수도 있다. 빠진 구성음수도 같고, 비화성음 개수도 같은 이 경우에는 C코드가 더 안정된 코드로 보고 Gsus4를 제거한다. 불안정한 코드로 간주되는 코드들은 sus4, aug, M7 코드 등이다.

(3) 그렇지 않다면 전위가 가장 낮은 것을 제외하고 나머지는 제거한다.

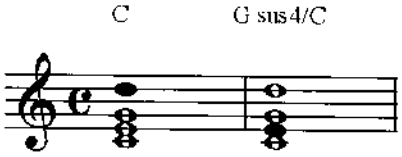

그림 5. 안정한 코드(C)와 불안정한 코드(Gsus4/C)
Fig. 5. Stable chord(C) and unstable chord(Gsus4/C).

2.7. 상황을 고려한 화음 제거

만일 구간 내에 두 개 이상의 화음들이 순차적으로 발생하는 경우에는 앞뒤 관계에 따라 적절하지 않은 화음을 제거할 필요가 있다. 인접한 화음의 모든 조합에 대하여 다음 경우를 적용한다:

(1) 좌측 코드가 불안정한 코드이고 우측 코드가 안정한 코드이면 우측 코드를 선택한다. 예를 들면, 그림 6과 같은 경우이다 (전타음으로 본다).

(2) 같은 베이스음을 공유하면 전위가 낮은 것을 선택한다. 예로 그림 7에 있는 경우에, Em를 선택한다.

(3) 위 두 경우를 처리하고 남아있는 코드들 중에서 같은 시간에 있는 코드들 중 가장 우선순위가 높은 것만을 선택한다.

(4) 그 결과 인접한 코드가 동일한 코드이면 통합한다.

(5) 만일 어떤 구간에 후보 코드가 하나도 없으나 오른쪽에 인접한 코드와의 다른 음이 1개 정도일 때 인접한 구간과 같은 구간으로 처리한다.

2.8. 후처리

이렇게 수정된 결과에 따라, 코드의 시작시간, 지속시간, 비화성음을 다시 계산한다. 이렇게 해서 얻어진 바흐 코랄 1번의 처리 결과의 예는 다음과 같다 (그림8):

(240, 120, 'C', 0, [2, 1, 1], 0, []),
(360, 120, 'G', 0, [2, 1, 1], 0, []),
(480, 120, 'Em', 0, [2, 1, 1], 0, [8]),
(600, 120, 'D', 1, [2, 1, 1], 0, []),
(720, 120, 'G', 0, [2, 1, 1], 0, []),
(840, 120, 'D', 0, [2, 1, 1], 0, [9]),
(960, 120, 'Em', 0, [1, 2, 1], 0, []),
(1080, 180, 'C', 0, [2, 1, 1], 0, [2, 11]),
(1260, 60, 'F#dim', 1, [1, 2, 1], 0, []),
(1320, 120, 'G', 0, [2, 1, 1], 0, []),
(1440, 240, 'D', 0, [2, 1, 1], 0, []),
...

그림 6. 불안정에서 안정으로 넘어가는 코드
Fig. 6. A chord from unstable state to stable.

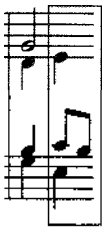

그림 7. 같은 베이스 음을 공유하는 경우
Fig. 7. Chords sharing a bass note.

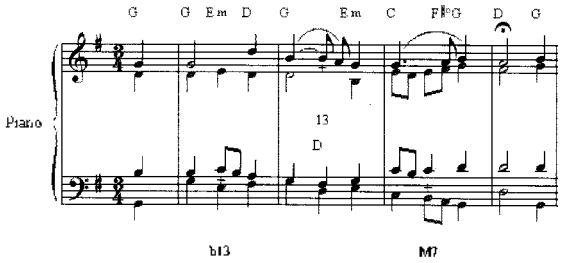

그림 8. 바흐 코랄 1번 처리 결과
Fig. 8. The result of Bach Choral No. 1.

III. 조성 결정 알고리듬

3.1 구성음에서 사용 음계를 이용한 후보 조성 구하기

일정 구간의 음들을 모아서 정리하면 사용된 후보 음계와 후보 조성을 얻을 수 있다. 반음계를 많이 사용하거나 조성감이 흐린 음악에서는 다소 적용하기 어렵지만, 사용된 음의 구성을 보면 대략 음계와 조성을 알 수 있다. 알고리듬 내에서 사용된 possible_Tonalities (stime, win size)이란 함수는 어떤 주어진 구간 (stime~stime + win_size) 내에서의 후보 조성을 구한다. 알고리듬은 다음과 같다.

1) 어떤 주어진 구간 내에 사용된 음을 모은다.
2) 중복되는 음들은 모두 제거한다. 옥타브 중복음도 제거한다 (집합 A).
3) 만일 출현 음들 (집합 A)이 어떤 음계 (집합 B)의 부분집합이면 1군 후보 조성 목록에 추가한다. 5단계로 이동한다. 그렇지 않으면 4단계로 이동한다.
4) 만일 차집합의 원소의 수 n(A - B)가 2군 후보의 차집합의 원소의 최소 수 보다 작으면 기존의 2군 후보를 버리고 음계 B를 2군 후보로 설정한다. 2군 후보가 없거나 2군 후보의 차집합의 원소의 최소 수가 같으면 2군 후보 목록에 추가한다.
5) 만일 1군 후보가 존재하면 2군 후보는 버리고 1군 후보를 조성 후보로 결정하고 리턴한다.
6) 그렇지 않으면, 2군 후보들의 조들이 관계조, (나란한조)이거나 같은 단조 군 (화성단음계, 가락단음계)이면 후보를 리턴하고 그렇지 않으면 후보 없음을 리턴한다.

3.2. 조성 변화 감지하기

음악은 언제든지 조가 변할 수 있기 때문에 세심한 검사가 필요하다. 조성 변화 감지를 위해 다음과 같은 가정을 한다. 1) 조는 최소한 한 마디 이상 지속된다. 2) 전조는 박 변화시점에서 이루어진다. 조의 지속 구간을 win_size 라 하고, 마디 크기를 measure_size, 박 길이를 beat size라 하면 초기 값으로 win size = measure_size로 설정하고 possible Tonalities (stime, win_size) 함수를 계속 불러, 같은 조를 리턴하는 한에는 measure_size를 계속 박 단위로 증가시킨다. 만일 다른 조를 리턴하면 이전 조성과 구간을 후보로 등록시키고 새로운 조성에 대해서도 변화가 일어날 때 까지 계속 구간을 증가시키면서 검사한다.

3.3. 조성 판단하기

후보 조성 목록이 구해지면 이제 앞서 계산된 화음 이름들이 후보 조성과 얼마나 잘 매치되는지 적합도를 측정하고 가장 적합도가 높은 조를 선택하게 된다. 적합도는 조와 코드의 발생 관계를 전수화한 것으로 다음 식 (1)로 표현된다.

$$F_T(\overline{C}) = \sum_{n=1}^{N} c_{dur}^{n} s_T(c^n) \tag{1}$$

여기서, $\overline{C} = c^1 c^2 c^3 ... c^N$ 이고, c^n은 2.9절의 결과로 나온 화음으로 c_{dur}^n은 화음의 지속 시간 (음가), $s_T(c^n)$는 화음 c^n가 T라는 조에 얼만큼 적합한지를 나타내는 수치이다. 예를 들어 C 장조에서 $\overline{C} = C, DT, GT, C$ 코드 진행이 생겼을 때 다음 식에 의해 계산된다.

$$F_{C_{maj}}(\overline{C}) = C_{dur} S_{C_{maj}}(C) + DT_{dur} S_{C_{maj}}(DT) + GT_{dur} S_{C_{maj}}(GT) + C_{dur} S_{C_{maj}}(C)$$

단, $0 \le s_T(c^n) \le 1.0$ 범위를 갖는다. C장조에 적합한 $S_{C_{maj}}(C)$, $S_{C_{maj}}(GT)$은 $S_{C_{maj}}(DT)$에 비해서 높은 점수를 갖는다. 동일한 구간 내에 이 적합도가 가장 높은 조가 최종적으로 선택된다. 점수는 경험적인 수치로 설정했다.

3.4. 구간 통합

앞서 계산된 조성은 미시적 관점에서 판단한 결과이다. 예를 들어 C -> D_7 -> G -> C로 진행했을 경우에 이것을 C장조로 볼 수도 있고 (I->II_7->V->I), G 장조로 볼 수도 있다 (IV->V_7 ->I->IV). 예를 들어 C (key C) -> D7 (key G) -> G -> C 로 분석되었다고 한다면 이것을 분리된 상태로 봐 들 수도 있고 C 장조 혹은 G 장조로 통합할 수도 있는데, 이에 대한 평가는 인접 구간의 두 조성을 따로 평가한 경우와 합쳐서 하나로 평가한 경우의 코드 적합도가 어느 쪽이 더 높은가를 판단해서 결정한다. 즉, 전체를 C조 혹은 G조로 놓고 화음 적합도를 계산한 결과 (score3)와 분리된 상태에서의 화음 적합도를 평가하되, 분리된 경우 (score1+score2)가 당연히 더 좋은 점수가 나올 것이기 때문에 (변화 화음을 조성 내의 화음으로 보기 때문) 통합된 경우는 변화화음으로 인해 적합도가 떨어지는 점을 고려해 약간의 이득 점수 (penalty)를 더 준다. 이 이득 점수 값도 경험적 수치 30을 적용했다. 다음과 같은 의사 코드로 형식화가 가능하다:

```
if score3+penalty > score1+score2:
    두 구간 통합
else:
    분리된 상태 유지
```

IV. 실험 및 고찰

실험은 개별적으로 입력된 테스트 코드들, 화성학 책에 있는 전조 예제들 [1], 바흐 코랄 중 1~6곡 [8], 찬송가 2곡, 기타 곡 6곡을 대상으로 진행되었다. Finale 2007에서 입력된 곡들은 MusicXML 파일 형식으로 변환되어 저장되었으며, 파이썬 언어 [9]로 제작된 프로그램은 이 MusicXML 파일을 읽고 코드 분석과 조성 분석을 수행하고 그 결과를 다시 MusicXML에 새로운 파일 이름으로 저장하였다. 이것을 다시 Finale 2007에서 읽어 들여 최종 출력물을 만들어 내었다. 그림 9에 입력 악보 형식을 보이며, 그림 10에 최종 출력 결과를 보인다. 이것은 출력 MusicXML 파일을 Finale 2007에서 읽어 들인 후에 화면 구성만 조절해서 출력한 것이다.

표시 형식에 대해 설명하면, 가장 위에 화음 이름(chord name)이 표시된다. 이것은 피날레의 코드 표시 부분이다. 출력 결과물에 추가의 정보를 출력하기 위해서 가사 (Lyrics)기능을 활용하였다. 1절에는 기본 조성과 상대적인 화성 및 전위 정보를 표기하고, 2절부터는 비화성음이 있을 경우에 그 종류를 쓴 것이다. 3절부터는 2절에 무슨 표기가 있을 때 겹치지 않도록 다음 절에 표기된다. 예를 들어 베이스 파트의 두 번째 음 G#은 아래에 M7으로 표기되어 근음으로부터 장7도 떨어진 비화성음임을 나타내었다. 5마디의 테너와 베이스가 모두 비화성음 처리되어 13, #11이라고 표기되었다. 이는 모두 근음으로부터의 거리이다.

실험대상으로 사용한 곡들에는 총 35회 (N)의 전조가 발생하는데 이 중에서 틀린 회수는 4회로 전조로 표기되어야 하는데 표기하지 못한 경우 (D) 2회, 다른 조성으로 잘 못 표기된 경우 (S)가 2회였다. 전조로 표기되지 않아야 하는데 표기된 경우 (I)는 한 경우도 발생하지 않았다. 따라서 올바른 처리율은 $1-\frac{(D+S+I)}{N}$ = 88.6% 가 되었다. 발생한 오류 중에 2회는 사람이 보기에도 다소 혼동을 일으킬 수 있는 부분으로 모호한 성격을 나타내었다. 오류는 모두 바흐 코랄에서 발생하였다.

그림 10의 결과에 알고리즘에 의해 처리된 것과 전문가에 의해서 분석된 결과를 보인다. 전문가에 의한 전조 분석은 가운데 큰 문자로 표기되었다. 결과에서 알 수 있듯이 전조가 대체로 잘 검출되었으나 마디1의 4번째 박의 E장조는 e단조로 보는 것이 타당하나 페르마타 부분에 i와 I가 같이 나오므로 혼동을 일으킬 수 있다고 판단된다. 만일 악절을 고려한다면 악절의 끝을 I 화음이 아닌 i# 화음으로 볼 수도 있을 것이다. 또 사람이 한 것과 약간의 차이를 보일 수 있는 부분은 두 번째 전조를 일으키는 마디 5의 Key=A의 위치가 마디 4의 마지막 박자로 옮겨 가는 것이 좋다. 이런 결과가 나온 이유는 구간 설정이 악절 (phrase)을 기준으로 하지 않고 있기 때문이다. 본 알고리즘은 악절 정보를 활용하고 있지 않은데 이 점은 앞으로 수정되어야 할 부분이다.

그림 11에 찬송가 406장의 처리 결과를 보여준다. 이 찬송가는 전조를 하고 있지는 않지만 부분적으로 이차 도

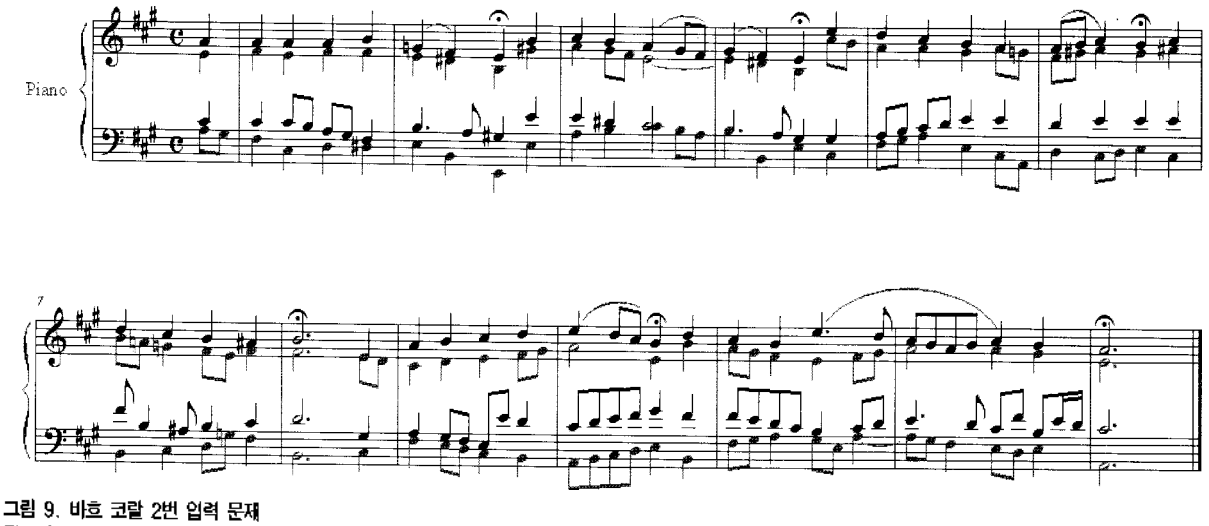

그림 9. 바흐 코랄 2번 입력 문제
Fig. 9. Bach Choral No. 2 input format.

미넌트 (secondary dominant) 코드가 사용되고 있고 (15, 16마디), 장식적 혹은 경과적 반음을 사용하고 있어서 (17, 18마디) 전조로 분류되었다가 다시 통합 과정을 거쳐서 단일 조로 복구된 경우이다.

그림 12의 원 부분은 C장조 완전 종지로 처리되어야 하는데 G 코드가 세컨더리 도미넌트 코드로 처리되었다. 이 부분은 원래 C장조로 분리 되었다가 통합된 부분으로 악절 (phrase) 종지에 따라서 우선적으로 C장조로 판단되어야 함에도 불구하고 그 대로 G장조에서 처리된 경우이다.

전체적인 결과에 다소간 미흡한 점이 있다면 조성 변화의 변화 시점을 하나의 시점이 아니라 공통코드 (pivot chord)를 중심으로 한 전조 구간을 표시하면 더 좋을 것으로 보인다. 또 중요한 문제는 악절 (phrase) 단위의 분석이 이루어지지 않는다는 것이다. 이것은 바흐 코랄 이외에는 악절을 구분할 만한 기호를 악보에서 찾기는 어렵고, 사용자가 직접 악절을 구분할 기호를 악보에 미리 입력해야 한다는 문제 때문에 제외시킨 것이지만 악절에 대한 구분을 받아들여서 전조 구분 기준점과 종지에 대한 처리해야 더 높은 정확도를 낼 수 있을 것으로 보인다

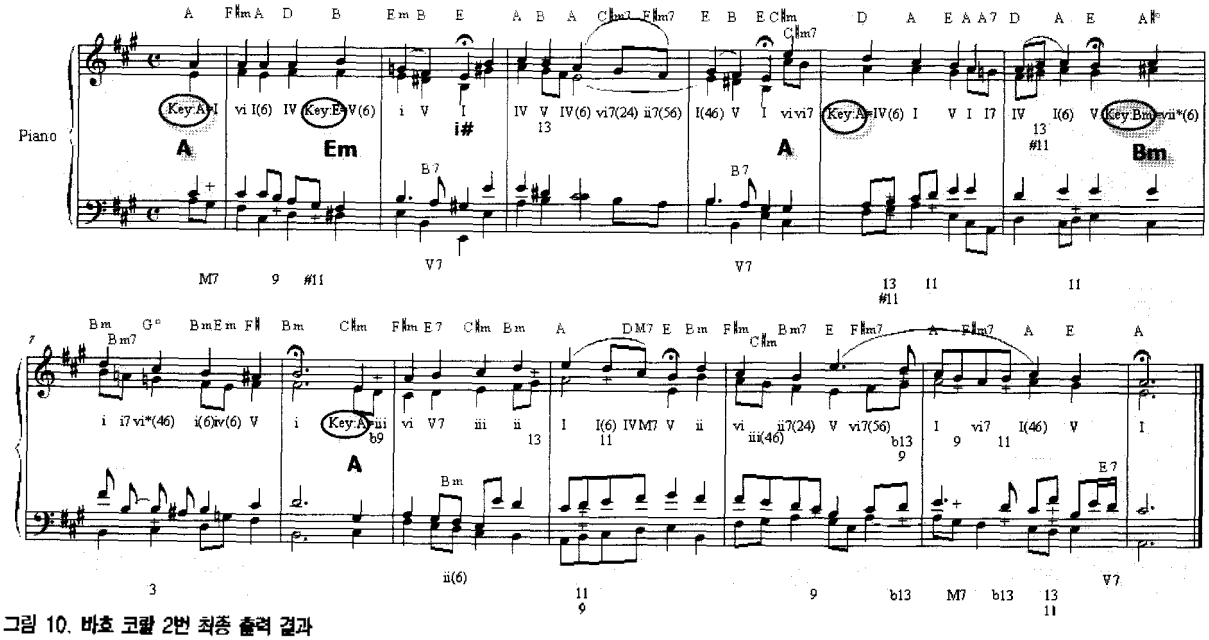

그림 10. 바흐 코랄 2번 최종 출력 결과
Fig. 10. Final result of Bach Choral No. 2.

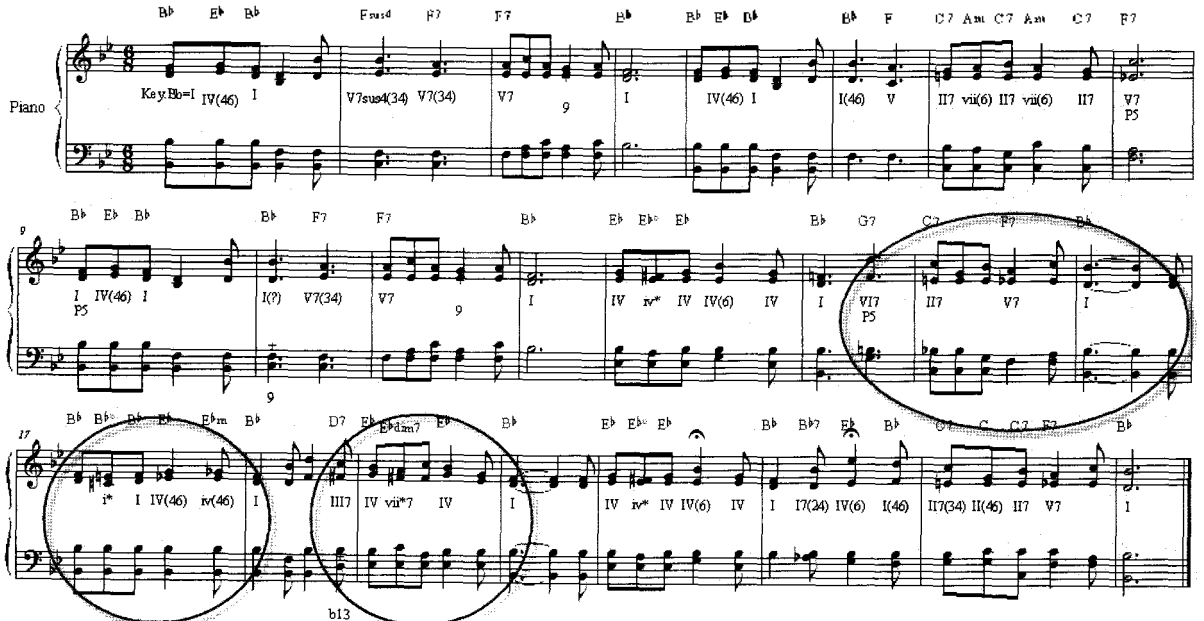

그림 11. 찬송가 406장-내 맘이 낙심되며, 원 부분은 부분 전조로 판단되었으나 구간 통합과정에서 병합된 결과임
Fig. 11. Hymn 406, Modulations are detected during the process but they are merged to have the same tonality.

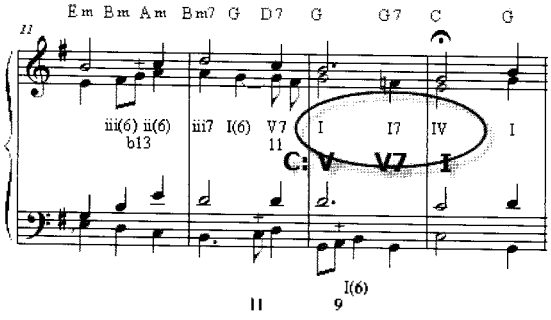

그림 12. 바흐 코랄 1번 중에 잘 못 처리된 경우 (C장조로 전조하고 정격 종지 처리 되어야 함)
Fig. 12. Wrong result of Bach Choral No. 1 (Authentic cadence must have been applied).

(오류중 2회). 또한 화음 표기를 기능화성 표기로 더 확장해야 하는 과제도 남겨두었다.

V. 결 론

본 논문은 MusicXML 파일로 표현되는 음악을 입력받아 화음 이름을 자동으로 판단하고, 사용된 음계와 검출된 화음 이름을 이용하여 조성을 검출하고, 조성과 화성이 표기된 최종 MusicXML 파일을 출력하는 알고리듬을 제시하였다. 최적의 화음 이름들을 검출하기 위하여 두 개 이상의 화음 이름으로 결정될 수 있는 여러 상황에서도 가장 적절한 화음을 선택하였고, 사용된 음들을 이용하여 음계를 추정하고, 구해진 화음 이름과 추정된 음계를 이용하여 음악의 조성을 파악하는 알고리듬을 기술하였다. 실험 결과 몇 경우를 제외한 대부분이 사람이 한 결과와 동일하게 나와 그 유효성을 보여주었다. 악절 단위의 분석을 해야 하는 것과 종지에 대한 적절한 처리 그리고 기능 화성의 표기 등의 보완이 추가로 요구되고 있다. 이 기술은 음악 분석을 통해서 음악을 손쉽게 체계화하고 정리할 수 있게 도와주어 음악 정보 검색과 같은 분야에 응용될 수 있게 하고, 화음 진행에 관한 통계를 이용하여 음악을 분류하거나 특징을 추출하는 분야에도 이용될 수 있을 것이다.

감사의 글

이 논문은 2006년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음

참 고 문 헌

1. 백병동, *화성학*, (수문당, 1984)
2. Temperley, D., "An Algorithm for Harmonic Analysis", Music Perception, 15/1, 31-68, 1997
3. Pardo, B. Birmingham, W.P., *Automated Partitioning of Tonal Music, Technical report*, (Electrical Engineering and Computer Science Department, University of Michigan, 1999)
4. Winograd, T., "Linguistic and Computer Analysis of Tonal Harmony", Journal of Music Theory, 12/1, 2-49, 1968
5. MusicXML, http://www.recordare.com/xml.html
6. Barthelemy Jerome, Bonardi Alain, *Figured bass and tonality recognition*. (ISMIR 2001, Bloomington, Oct, 2001)
7. 이강성, "화음 분석 구간 결정 알고리즘", 한국음향학회 춘계학술대회, 187, 2007.5.
8. 바흐, *4성코랄(371곡)*, (삼호출판사, 1989)
9. 이강성, *파이썬*, (프리렉, 2004)

저자 약력

• 이 강 성 (Gang Seong Lee)

1986년 2월: 광운대학교 컴퓨터공학과 학사
1988년 8월: 광운대학교 대학원 컴퓨터공학과 석사
1993년 2월: 광운대학교 대학원 컴퓨터공학과 박사
2006년 9월: 서울대학교 음악대학원 작곡과 석사과정
1994년 3월~현재: 광운대학교 교양학부 교수
1998년 8월~1999년 8월: Post Doc. Interactive Systems Laboratories, Carnegie Mellon University, U.S.A.

• 이 돈 응 (Donoung Lee)

1982년: 서울대학교 음악대학 졸업
1990년: 독일 프라이부르크 국립음대 졸업
1995년~2005년: 한양대학교 음악대학 교수
2005년~현재: 아시아작곡가 연맹 한국지부 회장
2005년~현재: 한국작곡가협회 부회장 2005년
2005년~현재: 서울대학교 음악대학 교수