(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2009-0083972
(43) 공개일자 2009년08월05일

(51) Int. Cl.
G06F 17/30 (2006.01)
(21) 출원번호 10-2008-0009865
(22) 출원일자 2008년01월31일
심사청구일자 2008년01월31일

(71) 출원인
연세대학교 산학협력단
서울 서대문구 신촌동 134 연세대학교
(72) 발명자
박상현
서울 양천구 신정동 327 목동아파트 1301동 1002호
유진희
서울 노원구 상계5동 한신3차아파트 12동 501호
(뒷면에 계속)
(74) 대리인
특허법인우인

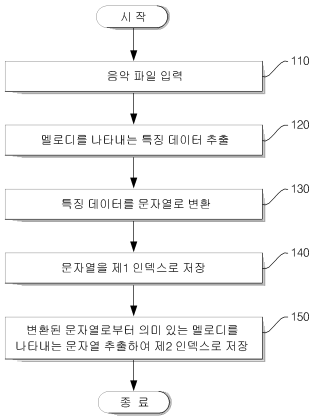
전체 청구항 수 : 총 18 항

(54) 음악 검색을 위한 음악 데이터베이스 구축 방법, 허밍 질의를 입력으로 하여 음악을 검색하는 방법 및 장치

(57) 요 약

음악 검색을 위한 음악 데이터이스 구축 방법, 허밍 질의를 입력으로 하여 음악을 검색하는 방법 및 장치가 개시된다. 본 발명에 따른 상기 기술적 과제를 해결하기 위하여 본 발명에 따른, 음악 검색을 위한 음악 데이터베이스 구축 방법은, 음악 파일을 입력받아 멜로디를 나타내는 특징 데이터를 추출하는 단계; 상기 추출된 특징 데이터를 문자열로 변환하는 단계; 상기 변환된 문자열을 검색의 기초가 되는 제1 인덱스로 저장하는 단계; 및 상기 변환된 문자열로부터 의미 있는 멜로디를 나타내는 문자열을 추출하여 검색의 기초가 되는 제2 인덱스로 저장하는 단계를 포함하는 것을 특징으로 한다. 이러한 본 발명에 의하면 허밍과 같은 부정확한 질의가 입력되더라도 검색의 정확도를 보장할 수 있고, 데이터베이스의 접근을 최소화함으로써 사용자가 찾고자 하는 음악을 검색하는 검색 속도를 향상시킬 수 있다.

대 표 도 - 도1



(72) 발명자

박치현

서울 은평구 갈현동 522-50

노홍찬

서울 강북구 수유4동 558-32호

특허청구의 범위

청구항 1

음악 검색을 위한 음악 데이터베이스 구축 방법에 있어서,

(a) 음악 파일을 입력받아 멜로디를 나타내는 특징 데이터를 추출하는 단계;

(b) 상기 추출된 특징 데이터를 문자열로 변환하는 단계;

(c) 상기 변환된 문자열을 검색의 기초가 되는 제1 인덱스로 저장하는 단계; 및

(d) 상기 변환된 문자열로부터 의미 있는 멜로디를 나타내는 문자열을 추출하여 검색의 기초가 되는 제2 인덱스로 저장하는 단계를 포함하는 것을 특징으로 하는 음악 데이터베이스 구축 방법.

청구항 2

제1항에 있어서,

상기 특징 데이터는 연속된 음표 간의 높이의 변화량과 길이의 변화율을 포함하는 것을 특징으로 하는 음악 데이터베이스 구축 방법.

청구항 3

제2항에 있어서,

상기 (b) 단계는, 상기 높이의 변화량과 상기 길이의 변화율의 조합을 소정 규칙에 따라 문자에 대응시킴으로써 상기 특징 데이터를 문자열로 변환하는 것을 특징으로 하는 음악 데이터베이스 구축 방법.

청구항 4

제3항에 있어서,

상기 (b) 단계는, 상기 높이의 변화량과 상기 길이의 변화율을 각 축으로 하는 2차원 공간을 복수 개의 구역으로 나누어, 동일한 구역에 속하는 상기 높이의 변화량과 상기 길이의 변화율의 조합을 동일한 문자에 대응시키는 것을 특징으로 하는 음악 데이터베이스 구축 방법.

청구항 5

제1항에 있어서,

상기 (c) 단계 및 상기 (d) 단계에서, 상기 문자열의 저장은 서픽스 트리(suffix tree) 형태로 저장하는 것을 특징으로 하는 음악 데이터베이스 구축 방법.

청구항 6

제1항에 있어서, 상기 (d) 단계는,

(d1) 상기 의미 있는 멜로디를 나타내는 문자열로서, 하나의 음악에서 빈번하게 발생되는 멜로디인 빈번 멜로디를 나타내는 문자열을 추출하여 검색의 기초가 되는 제2-1 인덱스로 저장하는 단계; 또는

(d2) 상기 의미 있는 멜로디를 나타내는 문자열로서, 소정 길이 이상의 쉼표를 경계로 나누어지는 멜로디를 나타내는 문자열을 추출하여 검색의 기초가 되는 제2-2 인덱스로 저장하는 단계를 포함하는 것을 특징으로 하는 음악 데이터베이스 구축 방법.

청구항 7

제6항에 있어서,

상기 (d1) 단계는,

(d11) 음악의 마디 단위로 문자열 패턴을 분석하여, 마디 단위의 문자열, 마디 위치, 발생 회수, 및 상기 발생 회수의 순위를 나타내는 리스트를 얻는 단계;

(d12) 상기 반복 회수의 순위가 가장 높은 마디들 각각을 대표 마디로 포함하는 그룹들을 생성하는 단계;

상기 생성된 각 그룹에 대하여, (d13) 상기 대표 마디와 인접하는 마디의 발생 회수를 소정 제1 임계값과 비교하고, 그 결과에 따라 상기 인접하는 마디를 그룹에 합치는 단계; 및 (d14) 상기 대표 마디의 발생 회수의 순위와 합쳐진 마디의 발생 회수의 순위의 차가 누적된 값을 소정 제2 임계값과 비교하고, 그 결과에 따라 상기 (d13) 단계를 반복하거나 종료하는 단계; 및

(d15) 상기 (d13) 단계 및 상기 (d14) 단계가 수행된 결과 얻어진 각 그룹에 포함되는 문자열을 상기 빈번 멜로디를 나타내는 문자열로서 추출하는 단계를 포함하는 것을 특징으로 하는 음악 데이터베이스 구축 방법.

청구항 8

제7항에 있어서, 상기 얻어진 각 그룹 간에 오버랩되는 문자열이 존재하는 경우, 해당하는 그룹들이 합쳐진 그룹에 포함되는 문자열을 상기 빈번 멜로디를 나타내는 문자열로서 추출하는 단계를 더 포함하는 것을 특징으로 하는 음악 데이터베이스 구축 방법.

청구항 9

허밍 질의를 입력으로 하여 음악을 검색하는 방법에 있어서,

(a) 허밍을 질의로서 입력받는 단계;

(b) 상기 입력받은 허밍으로부터 멜로디를 나타내는 특징 데이터를 추출하고, 상기 추출된 특징 데이터를 문자열로 변환하는 단계;

(c) 상기 변환된 문자열에 따라, 각종 음악의 멜로디를 나타내는 특징 데이터가 문자열로 변환되어 저장된 제1 인덱스 및 상기 변환된 문자열로부터 의미 있는 멜로디를 나타내는 문자열이 추출되어 저장된 제2 인덱스를 저장하고 있는 음악 데이터베이스를 이용하여 음악을 검색하는 단계를 포함하는 것을 특징으로 하는 음악 검색 방법.

청구항 10

제10항에 있어서, 상기 (b) 단계는,

상기 입력받은 허밍을 구성하는 음표의 높이와 길이를 추출하는 단계; 및

상기 추출된 음표의 높이와 길이로부터 연속된 음표 간의 높이의 변화량과 길이의 변화율을 포함하는 데이터를 상기 특징 데이터로서 추출하는 단계; 및

상기 추출된 특징 데이터를 문자열로 변환하는 단계를 포함하는 것을 특징으로 하는 음악 검색 방법.

청구항 11

제10항에 있어서,

상기 제2 인덱스는, 상기 의미 있는 멜로디를 나타내는 문자열로서, 빈번하게 발생되는 멜로디인 빈번 멜로디를 나타내는 문자열을 추출하여 저장된 제2-1 인덱스 또는 소정 길이 이상의 쉼표를 경계로 나누어지는 멜로디를 나타내는 문자열을 추출하여 저장된 제2-2 인덱스를 포함하는 것을 특징으로 하는 음악 검색 방법.

청구항 12

제11항에 있어서,

상기 (d) 단계는,

상기 변환된 문자열에 따라, 상기 제2-1 인덱스 및 상기 제2-2 인덱스로부터 음악을 검색하는 단계; 및

상기 변환된 문자열에 따라 상기 제1 인덱스로부터 음악을 검색하는 단계를 포함하는 것을 특징으로 하는 음악 검색 방법.

청구항 13

제12항에 있어서,

상기 음악 데이터베이스는 상기 각종 음악의 멜로디를 나타내는 특징 데이터를 더 저장하고 있고,

상기 (d) 단계는, 상기 (a) 단계에서 추출된 특징 데이터에 따라 상기 음악 데이터베이스에 저장된 특징 데이터로부터 음악을 검색하는 단계를 더 포함하는 것을 특징으로 하는 음악 검색 방법.

청구항 14

허밍 질의를 입력으로 하여 음악을 검색하는 장치에 있어서,

허밍을 입력받아, 멜로디를 나타내는 특징 데이터를 추출하고, 상기 추출된 특징 데이터를 문자열로 변환하는 질의 전처리부;

상기 검색을 위한 인덱스로서, 각종 음악의 멜로디를 나타내는 특징 데이터가 문자열로 변환되어 저장된 제1 인덱스 및 상기 변환된 문자열로부터 의미 있는 멜로디를 나타내는 문자열이 추출되어 저장된 제2 인덱스를 저장하고 있는 음악 데이터베이스; 및

상기 허밍으로부터 변환된 문자열에 따라, 상기 음악 데이터베이스로부터 음악을 검색하는 음악 검색부를 포함하는 것을 특징으로 하는 음악 검색 장치.

청구항 15

제14항에 있어서, 상기 질의 전처리부는 상기 입력받은 허밍을 구성하는 음표의 높이와 길이를 추출하고, 상기 추출된 음표의 높이와 길이로부터 연속된 음표 간의 높이의 변화량과 길이의 변화율을 포함하는 데이터를 상기 특징 데이터로서 추출하는 것을 특징으로 하는 음악 검색 장치.

청구항 16

제14항에 있어서,

상기 제2 인덱스는, 상기 의미 있는 멜로디를 나타내는 문자열로서, 빈번하게 발생되는 멜로디인 빈번 멜로디를 나타내는 문자열을 추출하여 저장된 제2-1 인덱스 또는 소정 길이 이상의 쉼표를 경계로 나누어지는 멜로디를 나타내는 문자열을 추출하여 저장된 제2-2 인덱스를 포함하는 것을 특징으로 하는 음악 검색 장치.

청구항 17

제16항에 있어서,

상기 음악 검색부는, 우선 상기 제2-1 인덱스 및 상기 제2-2 인덱스로부터 음악을 검색하고, 다음으로 상기 제1 인덱스로부터 음악을 검색하는 것을 특징으로 하는 음악 검색 장치.

청구항 18

제17항에 있어서,

상기 음악 데이터베이스는 상기 각종 음악의 멜로디를 나타내는 특징 데이터를 더 저장하고 있고,

상기 음악 검색부는, 상기 제1 인덱스로부터 음악을 검색한 이후에 상기 질의 전처리부에서 추출된 특징 데이터에 따라 상기 음악 데이터베이스에 저장된 특징 데이터로부터 음악을 검색하는 것을 특징으로 하는 음악 검색 장치.

명 세 서

발명의 상세한 설명

기 술 분 야

<1> 본 발명은 음악 검색에 관한 것으로, 보다 상세하게는 음악 검색을 위한 음악 데이터이스 구축 방법, 허밍 질의를 입력으로 하여 음악을 검색하는 방법 및 장치에 관한 것이다.

배 경 기 술

<2> 현재 내용 기반 음악 정보 검색 시스템에서 질의 특성에 따라 효과적인 특징 데이터를 찾는 연구가 활발히 진행

되고 있다. 이는 음표의 정확한 높이와 길이를 가진 원 음악과 같은 정확한 질의를 처리하기 위한 것과 허밍과 같은 부정확한 질의를 처리하기 위한 것으로 나뉘어진다.

<3> [1]에서 제안한 시스템에서 멜로디를 표현하기 위해 일반적으로 사용되는 특징 데이터로 'UDS' 문자열이 있는데 이는 인접한 두 음표의 높이를 사용하여 뒤의 음표가 앞의 음표보다 높으면 'U(Up)', 낮으면 'D(Down)', 같으면 'S(Same)' 로 표현한다. 그러나 이와 같은 데이터는 단지 음 높이의 곡선을 이용할 뿐 음의 길이는 무시한 정보이므로 정확도 측면에서 효율적이지 않다. 또 다른 허밍 질의 처리 시스템으로는 음악의 템포 (Tempo)를 사용하여 검색하는 방법이 있다[2]. 그러나 템포 데이터도 많은 특징 정보를 가지고 있지 않기 때문에 부정확한 허밍을 처리할 경우 정확도가 떨어지게 된다.

<4> 음악 정보 검색과 관련된 또 다른 연구 활동으로는 효율적인 인덱스 구조에 관한 것이 있다. 이러한 연구가 활발한 이유는 방대한 음악 데이터베이스를 순차적으로 검색함으로써 발생하는 오랜 검색 시간과 방대한 비교 계산 량을 줄이기 위해서이다. 이를 보완하기 위한 기존의 대표적인 인덱스 방법으로 N-gram이 있다[3]. 이 방법은 N개의 음표들을 하나의 단위로 취급하여 한 음표씩 오른쪽으로 이동시키며 인덱싱하는 방법으로 이는 크기가 N인 슬라이딩 윈도우 (Sliding Window)와 같은 개념이다. 이러한 인덱스 설계 방식으로 인하여 N-gram은 인덱스 크기가 크고 검색 비용 면에서 좋지 않은 성능을 보인다. 또한 검색 속도를 향상시키기 위해 숫자 형태의 특징 데이터를 문자열로 변환하지만 이는 부정확한 멜로디인 허밍의 특징을 고려하지 않은 문자열로서 질의가 부정확할수록 정확도가 급격히 떨어지게 된다. 또 다른 인덱스 방법으로는 공간 접근 방식 (Spatial Access Method)을 사용하는 경우가 있다[4]. 이 방법은 한 음표의 높이와 길이 정보를 이용하여 이를 2차원 공간상의 하나의 점으로 표현한 후 연속된 점들의 위치 정보를 인덱싱하는 방법이다. 대표적으로 R-Tree를 사용하는 방법이 있는데, 이는 검색 속도가 빠르지만 질의를 포함한 최소 영역 사각형 (Minimum Bounding Rectangle)안에 질의와 거리가 가까운 음악들이 많이 나타나기 때문에 정선도(Selectivity)가 좋지 않다는 단점을 가지고 있다.

<5> 음악 정보 검색에서 사용되는 내용 기반 인덱싱 기법은 음악의 특정 멜로디를 인덱싱하여 검색 속도를 줄이는 방법이 대부분이다. 이는 한 음악에서 어떤 부분의 멜로디를 인덱싱하느냐에 따라 음악 검색 시스템의 성능이 달라지기 때문에 성능을 향상시킬 수 있는 의미 있는 멜로디를 선택하는 것이 중요하다. 대표적으로 음악의 시작 부분을 인덱싱 함으로써 검색 속도를 향상시킨 방법이 있다[5]. 이는 사용자가 음악의 시작 부분을 자주 질의할 가능성이 높다는 것을 이용하였다. 그러나 이 방법은 사용자가 음악의 시작 부분을 질의하지 않을 경우 빠른 검색 속도를 보장하지 못한다. 또한 사용자가 음악의 시작 부분을 자주 질의한다는 것에 대한 충분한 근거를 제시하지 못하였다. 또 다른 내용 기반 인덱싱 방법으로는 일정하게 반복된 멜로디를 인덱싱한 연구가 있다[6]. 그러나 이 방법은 일정한 단위 없이 반복 멜로디를 찾기 때문에 반복 멜로디 추출 과정이 너무 복잡하고 후보 빈번 멜로디가 많이 발생하여 인덱스 크기가 커지는 단점을 가지고 있다. 앞의 두 방법과는 반대로 데이터베이스에 저장된 음악을 인덱싱하지 않고 사용자의 질의를 인덱싱하는 방법이 있다[7]. 이 방법은 사용자 질의 멜로디가 입력되면 이를 'UDS' 문자열로 변환한 후 질의들을 서로 비교하여 자주 질의되는 멜로디를 인덱싱하는 방법이다. 이는 사용자가 자주 질의한 음악은 빠르게 검색해 주는 장점을 가지고 있지만 만약 자주 질의하지 않는 멜로디가 입력이 되면 데이터베이스에 직접 접근해야 하는 단점이 있다. 또한 음의 박자를 고려하지 않고 음 높이의 곡선만을 이용한 'UDS' 문자열 방법을 사용하였기 때문에 정확도 측면에서 낮은 성능을 나타낸다.

<6> <참고 문헌>

<7> [1] N. Kosugi, Y. Nishihara, T. Sakata, M. Yamamuro and K. Kushima, "A Practical Query-By-Humming System for a Large Music Database", In Proc. ACM Multimedia, pp. 333-342, 2000.

<8> [2] E. Scheirer, "Tempo and Beat Analysis of Acoustic Musical Signals", Acoustic Society of America, 103:1 January, pp. 588-601, 1998.

<9> [3] A. Pienimaki, "Indexing Music Databases Using Automatic Extraction of Frequent Phrases", In Proc. 3rd International Symposium on Music Information Retrieval (ISMIR), pp. 25-30, 2002.

<10> [4] J. Reiss, J. Aucouturier and M. Sandler, "Efficient multidimensional searching routines for music information retrieval", In Proc. 2nd International Symposium on Music Information Retrieval (ISMIR),, pp. 163-171, 2001.

<11> [5] S. Doraisamy, S. Ruger, "Robust polyphonic music retrieval with n-grams", Journal of Intelligent Information Systems, 21:1, pp. 53-70, 2002.

<12> [6] C. Liu, J. Hsu and A. L. P. Chen, "Efficient Theme and Non-trivial Repeating Pattern

Discovering in Music Databases". In Proc. 15th International Conference on Data Engineering (ICDE '99), pp. 14-21, 1999.

<13> [7] 노승민, 박동문, 황인준, "사용자 질의 패턴을 이용한 효율적인 오디오 색인 기법", 한국 정보과학회 논문지, 제31권, 제1호, pp.143-153, 2004.

발명의 내용

해결 하고자하는 과제

<14> 본 발명이 이루고자 하는 기술적 과제는 허밍과 같은 부정확한 질의가 입력되더라도 검색의 정확도를 보장할 수 있고, 데이터베이스의 접근을 최소화함으로써 사용자가 찾고자 하는 음악을 검색하는 검색 속도를 향상시킬 수 있는, 음악 검색을 위한 음악 데이터이스 구축 방법을 제공하는 데 있다.

<15> 본 발명이 이루고자 하는 다른 기술적 과제는, 허밍과 같은 부정확한 질의가 입력되더라도 검색의 정확도를 보장할 수 있고, 데이터베이스의 접근을 최소화함으로써 검색 속도를 향상시킬 수 있는, 허밍 질의를 입력으로 하여 음악을 검색하는 방법 및 장치를 제공하는 데 있다.

과제 해결수단

<16> 상기 기술적 과제를 해결하기 위하여 본 발명에 따른, 음악 검색을 위한 음악 데이터베이스 구축 방법은, (a) 음악 파일을 입력받아 멜로디를 나타내는 특징 데이터를 추출하는 단계; (b) 상기 추출된 특징 데이터를 문자열로 변환하는 단계; (c) 상기 변환된 문자열을 검색의 기초가 되는 제1 인덱스로 저장하는 단계; 및 (d) 상기 변환된 문자열로부터 의미 있는 멜로디를 나타내는 문자열을 추출하여 검색의 기초가 되는 제2 인덱스로 저장하는 단계를 포함하는 것을 특징으로 한다.

<17> 여기서, 상기 특징 데이터는 연속된 음표 간의 높이의 변화량과 길이의 변화율을 포함하는 것이 바람직하다. 이때, 상기 (b) 단계는, 상기 높이의 변화량과 상기 길이의 변화율의 조합을 소정 규칙에 따라 문자에 대응시킴으로써 상기 특징 데이터를 문자열로 변환하는 것이 바람직하다. 또한, 상기 (b) 단계는, 상기 높이의 변화량과 상기 길이의 변화율을 각 축으로 하는 2차원 공간을 복수 개의 구역으로 나누어, 동일한 구역에 속하는 상기 높이의 변화량과 상기 길이의 변화율의 조합을 동일한 문자에 대응시키는 것이 바람직하다.

<18> 그리고, 상기 (c) 단계 및 상기 (d) 단계에서, 상기 문자열의 저장은 서픽스 트리(suffix tree) 형태로 저장하는 것이 바람직하다.

<19> 또한, 상기 (d) 단계는, (d1) 상기 의미 있는 멜로디를 나타내는 문자열로서, 하나의 음악에서 빈번하게 발생되는 멜로디인 빈번 멜로디를 나타내는 문자열을 추출하여 검색의 기초가 되는 제2-1 인덱스로 저장하는 단계; 또는 (d2) 상기 의미 있는 멜로디를 나타내는 문자열로서, 소정 길이 이상의 쉼표를 경계로 나누어지는 멜로디를 나타내는 문자열을 추출하여 검색의 기초가 되는 제2-2 인덱스로 저장하는 단계를 포함하는 것이 바람직하다.

<20> 이때, 상기 (d1) 단계는, (d11) 음악의 마디 단위로 문자열 패턴을 분석하여, 마디 단위의 문자열, 마디 위치, 발생 회수, 및 상기 발생 회수의 순위를 나타내는 리스트를 얻는 단계; (d12) 상기 반복 회수의 순위가 가장 높은 마디들 각각을 대표 마디로 포함하는 그룹들을 생성하는 단계; 상기 생성된 각 그룹에 대하여, (d13) 상기 대표 마디와 인접하는 마디의 발생 회수를 소정 제1 임계값과 비교하고, 그 결과에 따라 상기 인접하는 마디를 그룹에 합치는 단계; 및 (d14) 상기 대표 마디의 발생 회수의 순위와 합쳐진 마디의 발생 회수의 순위의 차가 누적된 값을 소정 제2 임계값과 비교하고, 그 결과에 따라 상기 (d13) 단계를 반복하거나 종료하는 단계; 및 (d15) 상기 (d13) 단계 및 상기 (d14) 단계가 수행된 결과 얻어진 각 그룹에 포함되는 문자열을 상기 빈번 멜로디를 나타내는 문자열로서 추출하는 단계를 포함할 수 있다.

<21> 나아가, 상기 얻어진 각 그룹 간에 오버랩되는 문자열이 존재하는 경우, 해당하는 그룹들이 합쳐진 그룹에 포함되는 문자열을 상기 빈번 멜로디를 나타내는 문자열로서 추출하는 단계를 더 포함할 수 있다.

<22> 상기 다른 기술적 과제를 해결하기 위하여, 본 발명에 따른 허밍 질의를 입력으로 하여 음악을 검색하는 방법은, (a) 허밍을 질의로서 입력받는 단계; (b) 상기 입력받은 허밍으로부터 멜로디를 나타내는 특징 데이터를 추출하고, 상기 추출된 특징 데이터를 문자열로 변환하는 단계; (c) 상기 변환된 문자열에 따라, 각종 음악의 멜로디를 나타내는 특징 데이터가 문자열로 변환되어 저장된 제1 인덱스 및 상기 변환된 문자열로부터 의미 있는 멜로디를 나타내는 문자열이 추출되어 저장된 제2 인덱스를 저장하고 있는 음악 데이터베이스를 이용하여

음악을 검색하는 단계를 포함하는 것을 특징으로 한다.

<23> 여기서, 상기 (b) 단계는, 상기 입력받은 허밍을 구성하는 음표의 높이와 길이를 추출하는 단계; 및 상기 추출된 음표의 높이와 길이로부터 연속된 음표 간의 높이의 변화량과 길이의 변화율을 포함하는 데이터를 상기 특징 데이터로서 추출하는 단계; 및 상기 추출된 특징 데이터를 문자열로 변환하는 단계를 포함하는 것이 바람직하다.

<24> 또한, 상기 제2 인덱스는, 상기 의미 있는 멜로디를 나타내는 문자열로서, 빈번하게 발생되는 멜로디인 빈번 멜로디를 나타내는 문자열을 추출하여 저장된 제2-1 인덱스 또는 소정 길이 이상의 쉼표를 경계로 나누어지는 멜로디를 나타내는 문자열을 추출하여 저장된 제2-2 인덱스를 포함하는 것이 바람직하다.

<25> 또한, 상기 (d) 단계는, 상기 변환된 문자열에 따라, 상기 제2-1 인덱스 및 상기 제2-2 인덱스로부터 음악을 검색하는 단계; 및 상기 변환된 문자열에 따라 상기 제1 인덱스로부터 음악을 검색하는 단계를 포함하는 것이 바람직하다.

<26> 또한, 상기 음악 데이터베이스는 상기 각종 음악의 멜로디를 나타내는 특징 데이터를 더 저장하고 있고, 상기 (d) 단계는, 상기 (a) 단계에서 추출된 특징 데이터에 따라 상기 음악 데이터베이스에 저장된 특징 데이터로부터 음악을 검색하는 단계를 더 포함할 수 있다.

<27> 상기 또 다른 기술적 과제를 해결하기 위하여 본 발명에 따른 허밍 질의를 입력으로 하여 음악을 검색하는 장치는, 허밍을 입력받아, 멜로디를 나타내는 특징 데이터를 추출하고, 상기 추출된 특징 데이터를 문자열로 변환하는 질의 전처리부; 상기 검색을 위한 인덱스로서, 각종 음악의 멜로디를 나타내는 특징 데이터가 문자열로 변환되어 저장된 제1 인덱스 및 상기 변환된 문자열로부터 의미 있는 멜로디를 나타내는 문자열이 추출되어 저장된 제2 인덱스를 저장하고 있는 음악 데이터베이스; 및 상기 허밍으로부터 변환된 문자열에 따라, 상기 음악 데이터베이스로부터 음악을 검색하는 음악 검색부를 포함하는 것을 특징으로 한다.

<28> 여기서, 상기 질의 전처리부는 상기 입력받은 허밍을 구성하는 음표의 높이와 길이를 추출하고, 상기 추출된 음표의 높이와 길이로부터 연속된 음표 간의 높이의 변화량과 길이의 변화율을 포함하는 데이터를 상기 특징 데이터로서 추출하는 것이 바람직하다.

<29> 또한, 상기 제2 인덱스는, 상기 의미 있는 멜로디를 나타내는 문자열로서, 빈번하게 발생되는 멜로디인 빈번 멜로디를 나타내는 문자열을 추출하여 저장된 제2-1 인덱스 또는 소정 길이 이상의 쉼표를 경계로 나누어지는 멜로디를 나타내는 문자열을 추출하여 저장된 제2-2 인덱스를 포함하는 것이 바람직하다.

<30> 또한, 상기 음악 검색부는, 우선 상기 제2-1 인덱스 및 상기 제2-2 인덱스로부터 음악을 검색하고, 다음으로 상기 제1 인덱스로부터 음악을 검색하는 것이 바람직하다.

<31> 또한, 상기 음악 데이터베이스는 상기 각종 음악의 멜로디를 나타내는 특징 데이터를 더 저장하고 있고, 상기 음악 검색부는, 상기 제1 인덱스로부터 음악을 검색한 이후에 상기 질의 전처리부에서 추출된 특징 데이터에 따라 상기 음악 데이터베이스에 저장된 특징 데이터로부터 음악을 검색할 수 있다.

효 과

<32> 상술한 본 발명에 의하면, 허밍과 같은 부정확한 질의가 입력되더라도 검색의 정확도를 보장할 수 있고, 데이터베이스의 접근을 최소화함으로써 사용자가 찾고자 하는 음악을 검색하는 검색 속도를 향상시킬 수 있다.

발명의 실시를 위한 구체적인 내용

<33> 이하에서는 도면을 참조하여 본 발명의 바람직한 실시예들을 상세히 설명한다. 이하 설명 및 첨부된 도면들에서 실질적으로 동일한 구성요소들은 각각 동일한 부호들로 나타냄으로써 중복 설명을 생략하기로 한다. 또한 본 발명을 설명함에 있어 관련된 공지기능 혹은 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그에 대한 상세한 설명은 생략하기로 한다.

<34> 도 1은 본 발명의 일 실시예에 따른 음악 검색을 위한 음악 데이터베이스 구축 방법의 흐름도이고, 도 2는 본 발명의 일 실시예에 따른 음악 데이터베이스의 구조도이다.

<35> 우선, 110단계에서 음악 파일을 입력받는다. 이때 음악 파일로는 미디(MIDI) 형태의 음악 파일을 입력받는 것이 바람직하다.

<36> 다음으로, 120단계에서, 입력받은 음악 파일로부터 멜로디를 나타내는 특징 데이터를 추출한다. 사용자의 질의를 입력으로 음악 검색을 하기 위해서는 음악 데이터베이스에 저장된 멜로디와 사용자의 질의 멜로디를 서로 비교하여 사용자가 원하는 음악을 검색하여야 하기 때문에, 음악 파일로부터 멜로디를 나타내는 특징 데이터를 추출하는 과정이 필요하다. 본 실시예에서, 멜로디를 나타내는 특징 데이터로서, 연속된 음표 또는 쉼표 간의 높이의 변화량과 길이의 변화율을 사용한다. 사용자가 입력한 허밍은 음표의 정확한 높이와 길이가 아닌 부정확한 멜로디이다. 다시 말해 사용자는 원 음악과 다르게 전체 음조를 높게 부르거나 낮게 부를 수 있고 원 음악보다 빠르게 부르거나 느리게 부를 수도 있다. 이러한 특징을 고려하여, 음악 검색의 정확도를 높이기 위하여 연속된 음표 간의 높이의 변화량과 길이의 변화율을 특징 데이터로 사용한다.

<37> 연속된 음표 또는 쉼표 간의 높이의 변화량과 길이의 변화량을 추출하기 위해서, 우선 음표의 높이와 길이, 쉼표의 길이, 마디 경계를 추출한다. 음표의 높이 정보를 위하여 미디 음악에서는 음의 높이를 미리 정의해 놓은 값으로서 옥타브 (Octave)를 사용한다. 옥타브의 표기는 알파벳 형태의 음계와 숫자로 이루어져 있고 값의 범위는 0부터 127까지 이다. 예를 들어 옥타브가 'F5'인 음표는 그 높이 값이 '65'이다. 본 실시예에서는 이러한 숫자 값을 사용하여 음표의 높이를 표현한다. 다음으로 음표와 쉼표의 길이를 추출한다. 미디 음악에서 음표의 길이는 4분 음표 즉, 한 박자의 길이(Time Base)를 기준으로 계산된다. 그러나 미디 음악마다 한 박자의 길이가 다를 수 있기 때문에 같은 음표라도 그 음의 길이는 음악마다 다른 값이 될 수 있다. 그러므로 모든 음악의 한 박자 길이를 동일한 값으로 변환해 주는 작업이 필요하다. 이와 같은 방법을 통해 모든 음표의 높이와 길이를 추출한다. 또한 이 단계에서 마디의 경계를 추출하는데 이를 추출하는 이유는 한 음악에서 빈번하게 발생하는 빈번 멜로디를 발견하고자 할때, 일정한 길이를 가진 단위로서 마디를 사용하기 위해서이다. 마디의 경계는 입력된 미디 음악의 전체 박자 즉, 몇 분의 몇 박자 정보(Time Signature)와 한 박자의 길이, 그리고 음표와 쉼표의 길이를 통해 계산이 가능하다.

<38> 음표 간의 높이 변화량이란 인접한 두 음표 사이에서 뒤의 음표의 높이와 앞의 음표의 높이의 차를 의미한다. 즉, 음표의 높이가 얼마만큼 증가, 감소했는지 또는 동일한지를 정확한 수치로 표현하는 것이다. 길이의 변화율은 인접한 두 음표 사이에서 뒤 음표의 길이를 앞 음표의 길이로 나누어 계산한다.

<39> 도 3은 본 발명의 일 실시예에 따라 연속된 음표 간의 높이의 변화량과 길이의 변화율을 추출하는 방법을 나타낸 개념도이다. 도 3을 참조하면, 두 개의 인접한 음표 사이에 높이의 변화량과 길이의 변화율을 성분으로 하는 2차원 벡터 형태의 특징 데이터를 추출하고 이 값을 앞 음표에 대한 정보로서 추출한다. 가령 인접한 두 음표를 N_i와 N_{i+1}이라고 하고, 각 음표의 높이를 P_i, P_{i+1}, 각 음표의 길이를 D_i, D_{i+1}이라 하면, N_i의 특징 데이터인 높이의 변화량(Pitch Interval)과 길이의 변화율(Duration Ratio)를 추출하기 위한 수식은 아래와 같다.

수학식 1

$$N_i = (P_i, D_i)$$
$$N_{i+1} = (P_{i+1}, D_{i+1})$$

$$Pitch\ Interval = P_{i+1} - P_i$$
$$Duration\ Ratio = D_{i+1} / D_i$$

<40>

<41> 도 4는 연속된 음표 사이에 쉼표가 존재하는 경우에 높이의 변화량과 길이의 변화율을 추출하는 방법을 나타낸 개념도이다. 종래의 연구에서는 쉼표를 무시한 경우가 대부분이지만 본 실시예에서는 사용자가 쉼표를 포함하여 불렀을 경우를 고려하여 쉼표 정보를 특징 데이터에 포함시킨다. 일반적으로 사용자는 일정한 길이 이상의 쉼표 후에 시작되는 멜로디를 질의할 가능성이 높기 때문에 쉼표의 위치가 중요하게 취급되어야 한다. 쉼표는 음표와는 달리 길이 정보만 가지고 있다. 따라서 도 4에 도시된 바와 같이 쉼표의 높이를 널 값 (Null)으로 저장하고 '#'으로 표현한다. 다음으로 쉼표의 특징 데이터를 추출하기 위해 높이의 변화량과 길이의 변화율을 계산한다. 이때 쉼표의 위치에 따라 두 가지의 경우로 나누어 처리할 수 있다. 첫 번째 경우는 음표 뒤에 쉼표가 나오는 경우이다. 이 때 계산되어야 할 특징 데이터는 앞 음표의 데이터가 되며 그 값은 도 4에 도시된 바와 같이 높이의 변화량은 변화가 없음을 의미하는 '0.00'으로 저장하고 길이의 변화율을 음표의 길이와 쉼표의 길이를 사용하여 '0.5'의 변화율로 계산함으로써 쉼표의 정보를 앞 음표의 정보에 포함시킨다. 이때 만일 쉼표 대신에 길이와 박자가 각각 67과 30인 음표가 이어진다면 위의 경우와 마찬가지로 (0.00, 0.5)라는 특징 데이터 값이 계산될 것이다. 이는 쉼표의 정보가 신뢰성이 없다는 문제점이 될 수 있지만 사용자가 허밍을 부를 경우 쉼표를 정

확히 지키지 못하고 음표로 인식하여 음을 계속 이어서 부를 경우가 발생할 수 있기 때문에 위와 같은 두 가지 경우를 분리하지 않고 생각한다. 다음으로 도 4에 도시된 바와 같이 쉼표 뒤에 음표가 나온다면 이 때 계산되어야 할 특징 데이터는 쉼표의 데이터가 되는데 이는 무시한다. 그 이유는 무시된 길이 변화율은 멜로디를 표현하지 못하는 무음의 멜로디이기 때문이다. 이러한 방법을 통해 하나의 미디 음악에서 연속된 2차원 벡터 형태의 특징 데이터들을 모두 생성한 후 이들을 도 2에 도시된 특징 데이터 저장부(210)에 저장한다.

<42> 다음으로, 130단계에서, 상기 120단계에서 추출된 특징 데이터를 문자열로 변환한다. 이미 설명한 바와 같이, 상기 120단계에서 추출된 특징 데이터는 연속된 음표 간의 높이의 변화량과 길이의 변화율을 성분으로 하는 2차원 벡터, 즉 높이의 변화량과 길이의 변화율의 조합이다. 본 실시예에서는 높이의 변화량과 길이의 변화율의 각 조합을 일정 규칙에 따라서 알파벳 문자에 대응시킴으로써 특징 데이터를 문자열로 변환한다.

<43> 사용자의 질의를 입력으로 음악 검색을 하는 경우, 2차원 벡터 형태로 특징 데이터가 저장된 데이터베이스를 검색한다면, 저장된 모든 음악을 순차적으로 비교하여 검색해야 한다. 이 때 데이터베이스에 저장된 2차원 벡터는 숫자 값으로 되어있고 검색을 위한 질의와의 비교 방법은 일반적으로 유클리디안 거리 (Euclidean Distance) 계산 방법을 사용한다. 그러므로 이러한 검색 방법을 사용하게 되면 데이터베이스의 크기가 증가될수록 검색 속도가 느려지게 된다. 이를 보완하기 위해 본 실시예에서는 특징 데이터를 문자열 형태로 변환하여 저장해 놓은 인덱스를 사용한다. 특징 데이터를 문자열로 변환하게 되면, 문자열 매칭 (String Matching) 알고리즘을 사용하여 음악 검색을 수행할 수 있고, 이 경우 숫자 형태로 이루어진 2차원 벡터를 비교하는 것보다 계산 복잡도가 훨씬 줄어들게 된다.

<44> 도 5는 본 발명의 일 실시예에 따라 높이의 변화량과 길이의 변화율을 성분으로 하는 2차원 벡터 값을 문자 값으로 변환하는 규칙을 나타낸 개념도이다. 도시된 바와 같이, 높이의 변화량과 길이의 변화율을 각 축으로 하는 2차원의 공간을 복수 개의 구간, 예를 들어 35개의 구간으로 나누고 각 구간을 하나의 문자에 할당하는 문자 변환 기준을 적용한다. 따라서 비슷한 2차원 벡터 값은 같은 문자로 치환되게 된다. 이는 허밍이 부정확하다는 특징을 반영한 것으로서 변환된 문자는 부정확한 허밍이 가지고 있는 에러를 허용한 값이 된다. 따라서 이러한 문자 변환 규칙을 적용함으로써 노래 실력이 없는 사람들이나 언어 장애를 가진 사람들도 효과적으로 음악을 검색할 수 있게 된다.

<45> 도 5를 참조하면, 가로축은 높이의 변화량을 의미하고 세로축은 길이의 변화율을 의미한다. 가로축에서 0값을 기준으로 오른쪽은 높이의 변화량이 양수 값을 보인다. 이는 인접한 두 음표 사이에서 뒤 음표의 높이가 앞 음표의 높이보다 높은 경우를 의미하고 오른 쪽으로 갈수록 그 차이는 점점 더 커짐을 의미한다. 반대로 가로축의 0값을 기준으로 왼쪽은 높이의 변화량이 음수 값을 보인다. 이는 뒤 음표의 높이가 앞 음표의 높이보다 낮음을 의미하고 왼쪽으로 갈수록 그 차이는 점점 커짐을 의미한다. 마찬가지로 세로축에서 1값을 기준으로 위쪽은 뒤 음표의 길이가 앞 음표의 길이보다 긴 경우이고 그 길이의 비율 차이는 위쪽으로 갈수록 점점 더 커짐을 의미한다. 반면 세로축의 1값을 기준으로 아래쪽은 뒤 음표의 길이가 앞 음표의 길이보다 짧은 경우이고 그 길이의 차이는 아래로 갈수록 점점 더 커짐을 의미한다. 도 5를 참조하면, -4와 +4 사이의 높이 변화량은 같은 문자로 변환되는데 이는 사용자가 실제 음악의 한 음을 부를 때 실제 음의 높이보다 4만큼 높게 부르거나 낮게 불러도 그만큼의 오차는 허용됨을 뜻한다. 마찬가지로 박자의 변화율에서 0.5와 2사이는 같은 문자로 변환된다. 이는 사용자가 실제 음악의 한 음을 부를 때 실제 길이보다 2배의 길이로 부르거나 0.5배의 길이로 불러도 그만큼의 오차를 허용된다는 의미이다.

<46> 도 6은 도 5에 도시된 바에 따른 변환 기준을 사용하여, 연속된 2차원 벡터 값의 연속을 문자열로 변환한 결과를 나타낸다.

<47> 다시 도 1을 참조하면, 140단계에서 상기 130단계에서 변환된 문자열을 제1 인덱스(220)로 저장한다. 이때 저장하는 형태는 서픽스 트리 (suffix tree) 형태로 저장하는 것이 바람직하다. 서픽스 트리란, 문자열의 내부 구조를 잘 드러내줄 수 있는 자료구조의 하나로서, 정확 매칭 알고리즘(exact matching algorithm)에 주로 응용된다.

<48> 다음으로, 150단계에서 상기 130단계에서 변환된 문자열로부터 의미 있는 멜로디를 나타내는 문자열을 추출하여 제2 인덱스로 저장한다. 한 음악은 여러 개의 멜로디로 이루어져 있는데, 멜로디란 음악적인 표현과 인간의 감정을 가장 잘 나타내는 요소로서 다양한 음높이와 길이 정보를 담고 있는 음표와 쉼표의 결합을 의미한다. 의미있는 멜로디란, 이와 같이 한 음악을 이루는 여러 멜로디 중 그 음악을 대표할 수 있는 멜로디, 즉 사용자가 질의할 가능성이 높은 멜로디를 말한다.

<49> 본 실시예에서 첫 번째 의미 있는 멜로디로서 한 음악에서 여러 번 반복되어 나타나는 임의의 길이를 가진 멜로디를 사용한다. 왜냐하면, 사용자가 음악을 들었을 경우 자주 반복하여 나타나는 멜로디를 기억하기 쉽기 때문이다. 즉, 사용자가 원하는 음악을 검색하기 위해서 그 음악의 멜로디들 중에 쉽게 기억할 수 있는 멜로디를 흥얼거릴 경우가 많기 마련이므로 본 실시예에서는 빈번하게 발생한 멜로디를 의미 있는 멜로디로 지정한다.

<50> 본 실시예에서 두 번째 의미 있는 멜로디로서 소정 길이 이상의 쉼표 사이에 있는 멜로디를 사용한다. 일정 길이 이상의 긴 쉼표는 멜로디를 나누는 경계로 볼 수 있다. 실제 음악 데이터를 분석하여 보면 멜로디의 흐름이 긴 쉼표를 경계로 나누어짐을 알 수 있다. 대부분의 음악에서 긴 쉼표는 시작 부분 전, 후렴 부분 전, 2절이 시작되기 전 등에 많이 나타난다. 즉, 음악의 후렴 부분과 처음 부분을 사용자가 기억하기 쉬울 것이라는 가정 하에 쉼표의 의미를 적용함으로써 소정 길이 이상의 쉼표로 나누어지는 멜로디를 의미 있는 멜로디로 지정한다.

<51> 먼저, 첫 번째 의미 있는 멜로디인 빈번 멜로디를 나타내는 문자열을 추출하는 과정을 설명한다. 도 7은 본 발명의 일 실시예에 따라 빈번 멜로디를 추출하는 방법을 나타낸 흐름도이다.

<52> 710단계에서 마디 단위로 문자열 패턴을 분석한다. 본 실시예에서, 한 음악에서 여러 번 반복되는 멜로디인 빈번 멜로디를 찾기 위해 우선 마디 단위로 문자열 패턴을 분석한다. 마디를 사용하여 빈번 멜로디를 추출하는 이유는, 마디란 동일한 박자를 가지고 있는 물리적이고 객관적인 단위이므로, 이를 기준으로 빈번 멜로디를 추출하며 정확하고 간결하게 빈번 멜로디를 얻을 수 있기 때문이다.

<53> 도 8은 마디 단위로 문자열 패턴을 분석하여 정보를 얻는 과정을 나타낸 개념도이다. 도 8에 도시된 바와 같이, 문자열 패턴 분석을 수행하기 위해 문자열 형태 나타내어진 하나의 음악을 마디 경계인 '/ ' 로 나눈다. 마디 단위로 나눈 문자열들을 문자열 패턴 매칭 알고리즘을 통해, 일치되는 문자열들을 분석하여 마디 단위의 문자열(Sequence in Bar), 마디의 위치 정보(이하, BarID), 각 문자열의 발생 회수, 즉 일치되는 마디들의 개수(이하, C), 그리고 C를 내림차순으로 정렬한 순위(이하, R)를 얻는다. 예를 들어 '그 후로 오랫동안'이란 제목을 가진 음악을 입력받아 문자열을 마디 단위로 나눈 후 마디 단위 문자열 패턴 분석을 수행하면 아래 표 1과 같은 리스트를 얻을 수 있다. 표 1을 참조하면, 문자열 <j, x, k, j, i, x>는 18, 22, 35, 39번째 마디에서 나타났고 발생 회수 C는 4임을 확인할 수 있다. 또한 이 문자열은 위의 음악에서 가장 많이 발생한 것이므로 순위 R은 1이 된다.

표 1

<54>

Sequence in Bar	Bar Position (BarId)	Count (C)	Rank (R)
<j, x, k, j, i, x>	18, 22, 35, 39	4	1
<k, k, j, x, g>	19, 36	2	2
<j, x, j, k, k, k>	20, 37	2	2
<k, j, j, w, g>	21, 38	2	2
…	…	…	…
<j, x, g, k, E, x, i, x>	17, 49	1	3

<55> 다시 도 7을 참조하면, 720단계에서 발생 회수의 순위가 가장 높은 마디, 즉 R=1인 마디들 각각을 대표 마디(Key Bar, 이하 KB)로 포함하는 그룹을 생성한다. 여기서, 그룹의 개수는 C개가 된다. 도 10은 표 1에 나타난 리스트를 기초로 그룹을 생성한 결과를 나타내는 도면이다. 도 10을 참조하면, C 값이 4이므로 4개의 그룹이 생성되었음을 알 수 있다. 도 10에서 수직선은 하나의 음악을 의미하고, 각 눈금은 하나의 마디를 의미한다. 또한 검은색이 칠해져 있는 눈금은 빈번 멜로디 생성을 위한 시작 마디인 각 그룹의 대표 마디를 의미한다. 도면 7과 같이 첫 번째 그룹의 대표 마디는 18번째 마디인 Bar18, 두 번째 그룹의 대표 마디는 Bar22, 세 번째 그룹의 대표 마디는 Bar35, 마지막으로 네 번째 그룹의 대표 마디는 Bar39이다.

<56> 본 실시예에서 빈번 멜로디를 추출하기 위해 두 가지 인자를 정의한다. 첫 번째 인자는 어떤 마디를 빈번 멜로디에 포함시킬지 결정하기 위한 선택 임계값으로서, BST(Bar Selection Threshold)라 부르기로 한다. 만일 BST 값이 2라면 발생 회수 C 값이 2 이상인 마디들만 사용하여 빈번 멜로디를 생성한다는 것을 의미한다. BST 값이 작을수록 빈번 멜로디의 길이가 길어질 수 있다. 따라서 두 번째 인자는 빈번 멜로디의 길이를 제한하기 위한 임계값으로서, 빈번 멜로디 길이 임계값(Frequent Melody Length Threshold, 이하 FMLT)이라 부르기로 한다. FMLT는 빈번 멜로디를 생성하면서 마디가 추가될 때마다 계산되는 변수(BSum)의 크기를 제한한다. 후술하겠지만, BSum은 대표 마디의 R 값과 합쳐진 마디의 R 값의 차를 계산하여 이전까지 계산된 BSum에 더하는

방식으로 계산한다. FMLT는 이러한 BSum의 상한을 결저하기 위한 값으로서, 본 실시예에서는 BSum이 FMLT 이하일 때까지 반복하여 빈번 멜로디를 생성해 나간다.

<57> 730단계에서는, 상기 720단계에서 문자열 패턴을 분석한 결과 얻어진 리스트, 상기 730단계에서 생성된 그룹 정보, BST 값, 그리고 FMLT 값 등을 이용하여, 각 그룹에 대해 대표 마디를 기준으로 인접한 마디를 합쳐 나가면서 빈번 멜로디를 생성한다.

<58> 도 9는 도 7에 도시된 730단계를 보다 구체적으로 나타낸 흐름도이다.

<59> 우선, 910단계에서 각 그룹을 나타내는 기호 c(1≤c≤C)를 c=1로 설정한다. 이는, 생성된 그룹들 중 첫 번째 그룹에서 후술하는 단계들이 시작됨을 의미한다.

<60> 이하 설명에서 대표 마디와 인접하는 양쪽 마디를 각각 KB_{right}, KB_{left}라 하기로 하고, 현재 마디라 부르기로 한다.

<61> 915단계에서 BSum의 초기값으로 BSum=0으로 설정하고, 920단계에서, BSum이 FMLT보다 크면 950단계로 진행하고, 그렇지 않으면 925단계로 진행한다.

<62> 925단계에서 현재 마디인 KB_{right}, KB_{left}의 C 값과 BST 값을 비교하여 C 값이 BST 값 이상인 마디가 있는 경우 후술하는 930단계 내지 940단계 중 어느 하나를 수행하고, 그렇지 않은 경우, 950단계로 진행한다.

<63> C 값이 BST 값 이상인 마디가 하나이면, 즉 KB_{right}과 KB_{left} 중 하나이면, 930단계에서, 해당하는 마디를 그룹에 합친다.

<64> C 값이 BST 값 이상인 마디가 두 개이고 각 C 값이 다르면, 935단계에서, C 값이 큰 마디를 그룹에 합친다.

<65> C 값이 BST 값 이상인 마디가 두 개이고 각 C 값이 동일하면, 940단계에서 두 마디를 모두 그룹에 합친다.

<66> 945단계에서, 상기 930단계, 935단계, 및 940단계 중 어느 한 단계에서 합쳐진 현재 마디의 R 값과 KB 마디의 R 값의 차를 BSum에 더한다. 그리고, 다시 상기 920 단계로 돌아가서, BSum 값과 FMLT 값의 비교 결과에 따라 상기 925단계 내지 945단계를 반복한다. 반복되는 경우에는, 현재 마디 중 오른쪽 마디가 합쳐졌다면 그 오른쪽 다음 마디인 $KB_{right+1}$을 현재 마디로 하여 상기 단계들을 수행하고, 현재 마디 중 왼쪽 마디가 합쳐졌다면 그 왼쪽의 다음 마디인 KB_{left-1}을 현재 마디로 보고 상기 단계들을 수행한다.

<67> 도 11은 BST 값과 FMLT 값이 각각 2와 6으로 주어진 상황에서 표 1에 나타난 리스트를 기초로, 도 10의 첫 번째 그룹에 대하여 빈번 멜로디를 생성하는 과정을 표현한 도면이다. 우선 첫 번째 그룹을 보면 대표 마디인 KB가 Bar18로 지정되어있다. 대표 마디의 양쪽 마디인 Bar17과 Bar19는 각각 KB_{left} 와 KB_{right}으로 할당되어진다. Bar17의 C값은 1로서 BST값보다 작기 때문에 조건을 만족시키지 못하므로 무시한다. 다시 말해 Bar17은 충분히 여러 번 반복되어 나타난 마디가 아니기 때문에 대표 마디인 Bar18과 합쳐질 수 없다. 반면 Bar19의 C값은 2로서 BST 값 이상이므로 Bar18과 합쳐진다. 마디를 합쳤다면 그 다음으로 그 그룹의 BSum을 계산한다. 합쳐진 Bar19의 R값은 2로서 대표 마디인 Bar18의 R값과의 차를 구하면 1이 된다. 이 값을 기존의 BSum값에 더하여 다시 BSum에 저장한다. 다음으로 Bar20의 C값을 BST값과 비교함으로써 위의 과정을 반복한다. 이때 대표 마디를 기준으로 왼쪽 방향은 Bar17의 C 값이 BST 값 이상이 되지 못하였으므로 빈번 멜로디를 생성하는데 무시된다. 오른쪽 방향으로 마디를 증가시켜가며 BSum이 FMLT를 넘지 않을 때까지 또는 C값이 BST값 보다 작은 마디가 나타날 때까지 빈번 멜로디를 생성하게 된다.

<68> BSum을 계산하기 위해 마디 개수인 C값을 사용할 수도 있지만 본 실시예에서는 C 값이 아닌 순위 R 값을 사용한다. 그 이유는 다음과 같다. 예를 들어 아래 표 2와 같은 정렬 순서를 가지는 두 음악을 고려하여 보자.

표 2

<69>

Music 1				Music 2			
Sequence in Bar	BarId	C	R	Sequence in Bar	BarId	C	R
c, c, a, d, d	6, 9, 20	3	1	a, e, e, s, d	4, 9, 15, 27, 30, 34	6	1
c, d, a, a, d	7, 21	2	2	c, e, s, a	5, 16, 28	3	2

a, d	8, 22	2	2	a, g, c, d	6, 10, 31	3	2
a, a, c, s	10, 23	2	2	e, s, c, c, d, s	3, 7, 11	3	2
b, e, h, z	11, 30	2	2	c, e, a	8, 12	3	2
b, b, k, u	12, 31	2	2	c, c, d	13, 14	3	2
c, c, a	13, 32	2	2	a, d, b, b	29, 31	3	2
...	...	...	...	...	...	...	...

<70> BSum을 계산하기 위해 R값 대신 C값을 사용한다고 가정한 후 각각 빈번 멜로디 생성 알고리즘을 적용해 보자. 만약 BST값은 2 그리고 FMLT값은 6으로 지정해 놓는다면 음악 1의 첫 번째 그룹은 6번째 마디부터 13번째 마디까지 8개의 마디로 이루어진 빈번 멜로디를 생성하게 된다. 이는 음악 1에서 빈번하게 발생한 마디들을 최대한 고려하여 BSum이 FMLT를 초과하지 않을 때까지 빈번 마디를 합쳐나감으로써 빈번 멜로디를 생성한 것이다. 또한 이는 순위 R값을 사용하여 BSum을 계산하여도 같은 빈번 멜로디가 생성됨을 알 수 있다. 반면, BSum을 계산하기 위해 C값을 사용하여 음악 2에서 첫 번째 그룹의 빈번 멜로디를 생성해보면 4번째 마디부터 6번째 마디까지 3개의 마디로 이루어진 빈번 멜로디를 얻을 수 있다. 그러나 7번째 마디 또한 음악 2에서는 빈번하게 나타난 마디임에도 불구하고 빈번 멜로디로 취급되지 않았다. 즉, 빈번하게 발생한 마디를 최대한 고려하지 못하기 때문에 부정확한 빈번 멜로디를 생성하게 된다. 반면 R값을 사용하여 빈번 멜로디를 생성하면 3번째 마디부터 10번째 마디까지 8개의 마디로 이루어진 빈번 멜로디를 생성할 수 있다. 이는 음악 2에서 빈번하게 발생한 마디들을 최대한 고려하여 빈번 멜로디를 생성한 결과라 할 수 있다. 이들 두 음악은 각각 다른 특성을 갖는 음악이므로 각 음악에 나타난 빈번 마디의 횟수가 다를 수 있다. 음악 1에서 가장 빈번한 마디 개수가 3인 반면 음악 2에서는 6이다. 그러나 이들 마디 개수의 순위 R은 1로 동일하다. 즉, 각각 다른 마디 개수를 가진 음악들의 빈번 멜로디를 생성하기 위해서는 그들의 마디 개수를 객관적인 수치로 표현하기 위해 순위를 적용하고, 이 값을 사용하여 BSum을 계산함으로써 빈번 멜로디를 생성해 나가는 것이다.

<71> 다시, 도 9를 참조하면, BSum이 FMLT 값보다 크게 되거나, 대표 마디의 양쪽 인접 마디 또는 합쳐진 마디의 인접 마디 중 C 값이 BST 값 이상인 마디가 없는 경우 그 그룹에 대하여는 빈번 멜로디의 생성이 종료되어야 한다. 따라서 이러한 경우, 이미 설명한 바와 같이 950단계로 진행하여 c를 1 증가시키고, 955단계에서 c와 그룹의 개수 C 값을 비교하여, c 가 그룹의 개수 C보다 크지 않으면, 상기 915단계로 진행하여 다른 그룹에 대하여 상기 915단계 내지 상기 945단계를 수행한다. 955단계에서 c가 그룹의 개수 C보다 크다면, 모든 그룹에 대하여 상기 915단계 내지 상기 945단계가 수행되었다는 의미가 된다. 그러면 960단계에서 C개의 그룹들 각각에 포함된 문자열들을 빈번 멜로디를 나타내는 문자열들로서 추출한다.

<72> 상술한 바와 같이, 모든 그룹에서 빈번 멜로디 생성이 완료되면 각 그룹 안에는 빈번하게 발생한 마디들로 이루어진 빈번 멜로디가 생성되어 있을 것이다. 그러나 도 12에 도시된 바와 같이 두 개의 그룹에 속한 빈번 멜로디가 오버랩되는 상황이 발생할 수 있다. 이때 각 그룹 안에 있는 빈번 멜로디를 각각 따로 저장한다면 중복된 데이터가 많이 발생할 것이다. 그러므로 이를 막기 위해 오버랩되는 그룹들을 하나의 그룹으로 합침으로써 중복된 멜로디를 하나의 멜로디로 취급할 수 있다. 따라서 도 9에는 각 그룹 간에 오버랩되는 문자열이 존재하는 경우, 해당하는 그룹들이 하나로 합쳐진 그룹에 포함되는 문자열을 빈번 멜로디를 나타내는 문자열로 추출하는 단계를 더 포함시킬 수 있다. 도 13은 이러한 실시예에 따라 표 1에 나타난 리스트를 기초로 최종적으로 빈번 멜로디를 나타내는 문자열이 추출된 결과를 나타낸다. 첫 번째 그룹과 두 번째 그룹은 상당한 부분이 오버랩되므로 하나의 그룹으로 합쳐졌으며, 세 번째 그룹과 네 번째 그룹 역시 마찬가지로 하나의 그룹으로 합쳐졌다. 따라서 최종적으로 2개의 그룹이 생성되고, 이들 그룹에 포함된 문자열을 빈번 멜로디를 나타내는 문자열로 취급한다.

<73> 상술한 과정을 통하여 생성된 빈번 멜로디를 나타내는 문자열은 도 2에 도시된 제2-1 인덱스(230)에 저장한다. 여기서 문자열을 저장하는 형태는 제1 인덱스(220)와 마찬가지로, 서픽스 트리 형태로 저장하는 것이 바람직하다.

<74> 이제, 두 번째 의미 있는 멜로디인 소정 길이 이상의 쉼표를 경계로 나누어지는 멜로디를 나타내는 문자열을 추출하는 과정을 설명한다. 이를 위하여 상술한 도 1의 120단계, 즉 멜로디를 나타내는 특징 데이터를 추출하는 과정에서 소정 길이 이상의 쉼표의 위치 정보를 따로 저장해 놓는다. 그리고 그 위치를 경계로 하여 음악을 나눔으로써 여러 개의 쉼표로 나누어진 멜로디를 생성한다. 소정 길이 이상의 쉼표란 멜로디의 연속된 흐름이 끊어지는 부분으로서, 소정 길이를 어떤 값으로 정의하느냐에 따라 쉼표를 경계로 나누어지는 멜로디의 길이가 달

라질 수 있다. 예를 들어 한 박자, 즉 4분 쉼표를 기준으로 멜로디를 나눈다면, 한 박자 이상의 길이를 갖는 쉼표가 나타난 곳을 경계로 멜로디를 나눈다.

<75> 도 14는 짧은 멜로디를 한 박자 이상의 쉼표로 나눈 결과를 나타내는 도면이다. 특징 데이터를 추출하는 과정에서, 한 박자의 쉼표는 앞 음표의 특징 데이터를 생성하는데 사용되고, 쉼표의 특징 데이터는 무시되면서 단지 그 위치 정보만 저장된다. 그리고 문자 형태로 변환한 후 저장해 놓은 쉼표 위치를 경계로 멜로디를 나눈다. 도시된 바와 같이 4분 쉼표를 경계로 문자열 [x,w,x]와 [x]가 각각 생성되었음을 알 수 있다.

<76> 상술한 과정을 통하여 생성된 빈번 멜로디를 나타내는 문자열은 도 2에 도시된 제2-2 인덱스(240)에 저장한다. 여기서 문자열을 저장하는 형태는 제1 인덱스(220) 및 제2-1 인덱스(230)와 마찬가지로, 서픽스 트리 형태로 저장하는 것이 바람직하다.

<77> 도 15는 상술한 본 발명의 실시예에 따라서 모든 멜로디를 나타내는 문자열, 빈번 멜로디를 나타내는 문자열, 쉼표를 기준으로 나누어진 멜로디를 나타내는 문자열을 추출하여 저장하는 과정을 나타낸 개념도이다. 도시된 바와 같이, 먼저, 음악 전체를 나타내는 문자열을 서픽스 트리 형태로 제1 인덱스로 저장하고, 음악 전체를 나타내는 문자열로부터 빈번 멜로디를 나타내는 문자열을 추출하고, 제2-1 인덱스로 저장하며, 음악 전체를 나타내는 문자열로부터 쉼표를 기준으로 나누어진 멜로디를 추출하여 제2-2 인덱스로 저장한다. 서픽스 트리를 사용하게 되면 사용자가 서픽스 트리에 저장된 멜로디 안의 임의의 부분부터 질의하여도 검색이 가능하게 된다. 허밍 질의를 입력으로 도 15에 도시된 바에 따라 구축된 음악 데이터베이스를 통하여 음악 검색을 하는 경우, 우선 제2-1 인덱스 및 제2-2 인덱스를 먼저 검색하고, 그 다음에 제1 인덱스를 검색하면 사용자가 원하는 음악을 검색하는 데 걸리는 시간을 줄일 수 있게 된다.

<78> 도 16은 본 발명의 일 실시예에 따른 허밍 질의를 입력으로 하여 음악을 검색하는 장치의 블록도이다. 도 16을 참조하면, 본 실시예에 따른 음악 검색 장치는 질의 전처리부(1610), 검색부(1620), 디스플레이(1630)부, 음악 데이터베이스(1640)를 포함하여 이루어진다.

<79> 음악 데이터베이스(1640)에는 상술한 음악 데이터베이스 구축 방법에 따라 도 2에 도시된 바와 마찬가지로 데이터가 저장되어 있다. 즉, 음악 데이터베이스(1640)는 각종 음악의 멜로디를 나타내는 특징 데이터를 저장하는 특징 데이터 저장부(210), 특징 데이터가 문자열로 변환되어 저장된 제1 인덱스(220), 의미 있는 멜로디로서 빈번 멜로디 및 소정 길이 이상의 쉼표를 경계로 나누어지는 멜로디를 각각 저장하는 제2-1 인덱스(230) 및 제2-2 인덱스(240)를 포함한다.

<80> 질의 전처리부(1610)는 사용자로부터 허밍을 질의로서 입력받아, 멜로디를 나타내는 특징 데이터를 추출하고, 추출된 특징 데이터를 문자열로 변환하는 질의 전처리 과정을 수행한다. 입력받은 질의는 사용자가 마이크를 통해 흥얼거린 허밍이다. 허밍 데이터는 상술한 음악 데이터베이스(1640) 구축을 위해 입력되는 형태인 미디와 달리 웨이브(wave) 형태를 지니고 있다. 따라서, 질의 전처리부(1610)는 우선 웨이브 형태의 허밍 데이터에서 미디 형태에 들어 있는 정보와 같이 음의 높이와 길이, 그리고 쉼표의 높이를 추출한다. 그리고, 이렇게 추출된 정보로부터 특징 데이터 저장부(210)에 저장된 각종 음악의 특징 데이터와 같은 형태로 특징 데이터를 추출한다. 즉, 음표 또는 쉼표의 높이의 변화량과 길이의 변화율을 계산한다. 다음으로, 질의 전처리부(1610)는 허밍의 특징 데이터를 문자열 형태로 변환한다. 이때 물론, 특징 데이터베이스의 구축 과정에서 각종 음악의 특징 데이터가 문자열로 변환되는 기준과 동일한 기준을 사용한다.

<81> 검색부(1620)는 질의 전처리부(1610)에서 허밍으로부터 추출된 문자열을 가지고 음악 데이터베이스(1640)에 저장된 각종 음악의 제1 인덱스(220), 제2-1 인덱스(220), 제2-2 인덱스(240) 또는 특징 데이터로부터 음악을 검색한다. 이때 검색부(1620)는 문자열이 서픽스 트리 형태로 저장된 각 인덱스로부터 음악을 검색하는 경우 문자열의 비교 방법으로 정확 매칭 알고리즘(Exact Matching Algorithm)을 사용할 수 있다.

<82> 그리고, 디스플레이(1630)는 검색부(1620)에서 검색된 결과를 사용자가 알 수 있도록 표시한다. 이때 예를 들어 검색된 음악의 곡명, 가수 등을 검색 결과로서 표시할 수 있다.

<83> 도 17은 본 발명의 일 실시예에 따른 허밍 질의를 입력으로 하여 음악을 검색하는 방법의 흐름도이다. 본 실시예에 따른 음악 검색 방법은 도 16에 도시된 음악 검색 장치에서 수행된다.

<84> 음악 검색 장치는 사용자로부터 허밍을 질의로서 입력받는다(1710단계).

<85> 질의 전처리부(1610)는 입력받은 허밍에 대하여 상술한 바와 같은 질의 전처리 과정을 수행한다(1715단계).

<86> 검색부(1620)는 질의 전처리 과정을 통해 변환된 허밍 데이터에 따라 제2-1 인덱스로부터 음악을 검색하고(1720

단계), 이와 병렬적으로 제2-2 인덱스로부터 음악을 검색한다(1725단계) .

<87> 검색이 성공하면(1730단계), 검색 결과를 디스플레이(1630)를 통하여 출력한다(1735단계). 사용자는 출력되는 검색 결과를 보고 자신이 원하는 음악이 검색되었으면 음악 검색 장치에 마련된 사용자 인터페이스(미도시)를 통하여 검색을 중지시킬 수 있다. 따라서 사용자로부터 검색 중지 명령이 입력되면(1780단계), 검색을 종료한다.

<88> 1730단계에서 검색이 실패하면, 혹은 검색 결과가 나오더라도 사용자로부터 검색 중지가 요청되지 않으면 검색부(1620)는 질의 전처리 과정을 통해 변환된 허밍 데이터에 따라 제1 인덱스로부터 음악을 검색한다(1740단계).

<89> 검색이 성공하면(1745단계에서), 검색 결과를 디스플레이(1630)를 통하여 출력한다(1750단계). 상술한 제2-1 및 2-2 인덱스로부터의 검색과 마찬가지로, 사용자는 출력되는 검색 결과를 보고 자신이 원하는 음악이 검색되었으면 검색을 중지시킬 수 있다. 따라서, 제1 인덱스로부터의 검색 중에도 사용자로부터 검색 중지 명령이 입력되면(1780단계) 검색을 종료한다.

<90> 제1 인덱스로부터도 검색 결과가 나오지 않거나, 사용자로부터 검색 중지가 요청되지 않으면, 검색부(1620)는 질의 전처리부(1610)에서 허밍 데이터가 변화된 숫자 형태의 특징 데이터를 가지고, 음악 데이터베이스(1640)에 저장된 특징 데이터로부터 음악을 검색한다(1755단계). 여기서, 허밍의 특징 데이터와 음악 데이터베이스(1640)에 저장된 특징 데이터와의 비교 방법으로는, 유클리디안 거리 함수를 사용하여 가까운 거리의 음악을 얻어내는 방법을 사용할 수 있다. 검색이 성공하면(1760단계), 검색 결과를 디스플레이(1630)를 통하여 출력하고(1770단계), 검색이 실패하면 디스플레이(1630)를 통하여 검색 실패를 알리는 메시지를 출력한다(1765단계). 특징 데이터로부터의 검색 중에도 사용자로부터 검색 중지 명령이 입력되면(1780단계), 검색을 종료한다.

<91> 상술한 바와 같이, 본 실시예에 의하면, 우선 빈번 멜로디 및 쉼표 단위의 멜로디를 나타내는 문자열이 저장된 제2-1 인덱스 및 제2-2 인덱스로부터 검색을 수행한다(제1단계 검색). 따라서 사용자가 빈번 멜로디를 흥얼거렸거나, 일정 길이 이상의 쉼표 다음에 시작하는 멜로디를 흥얼거렸다면 제1단계 검색에서 검색이 완료되어 사용자에게 가장 빠른 시간 내에 검색 결과를 전달할 수 있다. 제1단계 검색에서 사용자가 원하는 검색 결과를 얻지 못했다면 음악 전체를 나타내는 문자열이 저장된 제1 인덱스로부터 검색을 수행한다(제2단계 검색). 제2단계 검색에 의해, 사용자가 빈번 멜로디 또는 쉼표 다음에 시작하는 멜로디를 질의하지 않더라도 검색이 완료될 수 있다. 물론, 제2단계 검색에서도 사용자가 원하는 검색 결과가 나오지 않을 수 있으므로, 마지막으로 숫자 형태로 저장된 특징 데이터로부터 음악을 검색한다(제3단계 검색). 숫자 형태의 특징 데이터는 음악을 가장 정확하게 표현하는 데이터이므로, 제3단계 검색에서 가장 정확한 검색이 이루어진다. 이처럼 본 실시예에 의하면, 사용자가 입력할 가능성이 가장 높은 질의를 바탕으로 검색하는 제1단계 검색을 먼저 수행한다. 또한, 상대적으로 검색 속도가 빠른 제1단계 검색 및 제2단계 검색을 먼저 수행하고, 나중에 제3단계 검색을 수행한다. 따라서 사용자가 원하는 음악을 검색하는 데 걸리는 시간을 최소화할 수 있으며, 검색 시간이 오래 걸리는 특징 데이터에의 접근을 최소화할 수 있다.

<92> 한편, 상술한 본 발명의 실시예들은 컴퓨터에서 실행될 수 있는 프로그램으로 작성가능하고, 컴퓨터로 읽을 수 있는 기록매체를 이용하여 상기 프로그램을 동작시키는 범용 디지털 컴퓨터에서 구현될 수 있다. 상기 컴퓨터로 읽을 수 있는 기록매체는 마그네틱 저장매체(예를 들면, 롬, 플로피 디스크, 하드 디스크 등), 광학적 판독 매체(예를 들면, 시디롬, 디브이디 등) 및 캐리어 웨이브(예를 들면, 인터넷을 통한 전송)와 같은 저장매체를 포함한다.

<93> 이제까지 본 발명에 대하여 그 바람직한 실시예들을 중심으로 살펴보았다. 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

도면의 간단한 설명

<94> 도 1은 본 발명의 일 실시예에 따른 음악 검색을 위한 음악 데이터베이스 구축 방법의 흐름도이다.

<95> 도 2는 본 발명의 일 실시예에 따른 음악 데이터베이스의 구조도이다.

<96> 도 3은 본 발명의 일 실시예에 따라 연속된 음표 간의 높이의 변화량과 길이의 변화율을 추출하는 방법을 나타낸 개념도이다.

<97> 도 4는 연속된 음표 사이에 쉼표가 존재하는 경우에 높이의 변화량과 길이의 변화율을 추출하는 방법을 나타낸 개념도이다.

<98> 도 5는 본 발명의 일 실시예에 따라 높이의 변화량과 길이의 변화율을 성분으로 하는 2차원 벡터 값을 문자 값으로 변환하는 규칙을 나타낸 개념도이다.

<99> 도 6은 도 5에 도시된 바에 따른 변환 기준을 사용하여, 연속된 2차원 벡터 값의 연속을 문자열로 변환한 결과를 나타낸다.

<100> 도 7은 본 발명의 일 실시예에 따라 빈번 멜로디를 추출하는 방법을 나타낸 흐름도이다.

<101> 도 8은 마디 단위로 문자열 패턴을 분석하여 정보를 얻는 과정을 나타낸 개념도이다.

<102> 도 9는 도 7에 도시된 730단계를 보다 구체적으로 나타낸 흐름도이다.

<103> 도 10은 표 1에 나타난 리스트를 기초로 그룹을 생성한 결과를 나타내는 도면이다.

<104> 도 11은 표 1에 나타난 리스트를 기초로, 도 10의 첫 번째 그룹에 대하여 빈번 멜로디를 생성하는 과정을 표현한 도면이다.

<105> 도 12는 두 개의 그룹에 속한 빈번 멜로디가 오버랩되는 상황을 설명하기 위한 개념도이다.

<106> 도 13은 표 1에 나타난 리스트를 기초로 최종적으로 빈번 멜로디를 나타내는 문자열이 추출된 결과를 나타낸다.

<107> 도 14는 짧은 멜로디를 한 박자 이상의 쉼표로 나눈 결과를 나타내는 도면이다.

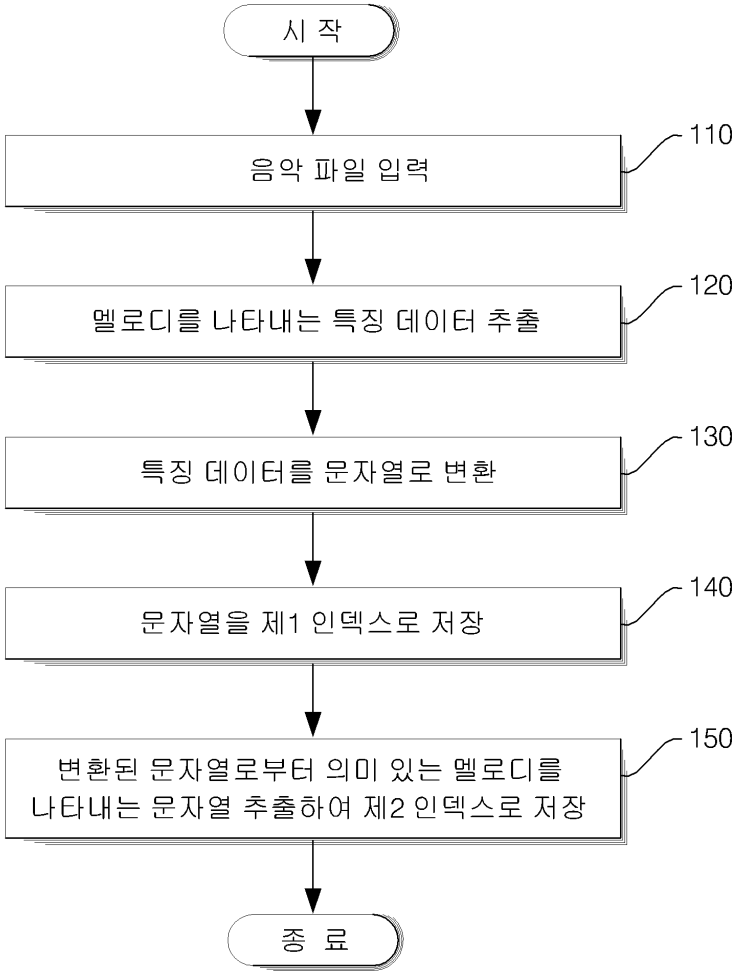
<108> 도 15는 모든 멜로디를 나타내는 문자열, 빈번 멜로디를 나타내는 문자열, 쉼표를 기준으로 나누어진 멜로디를 나타내는 문자열을 추출하여 저장하는 과정을 나타낸 개념도이다.

<109> 도 16은 본 발명의 일 실시예에 따른 허밍 질의를 입력으로 하여 음악을 검색하는 장치의 블록도이다.

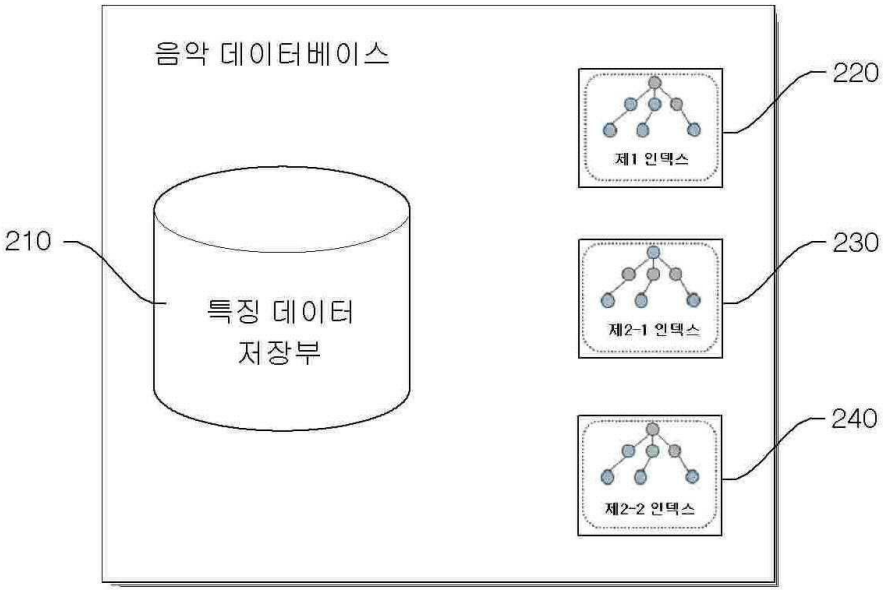
<110> 도 17은 본 발명의 일 실시예에 따른 허밍 질의를 입력으로 하여 음악을 검색하는 방법의 흐름도이다.

도면

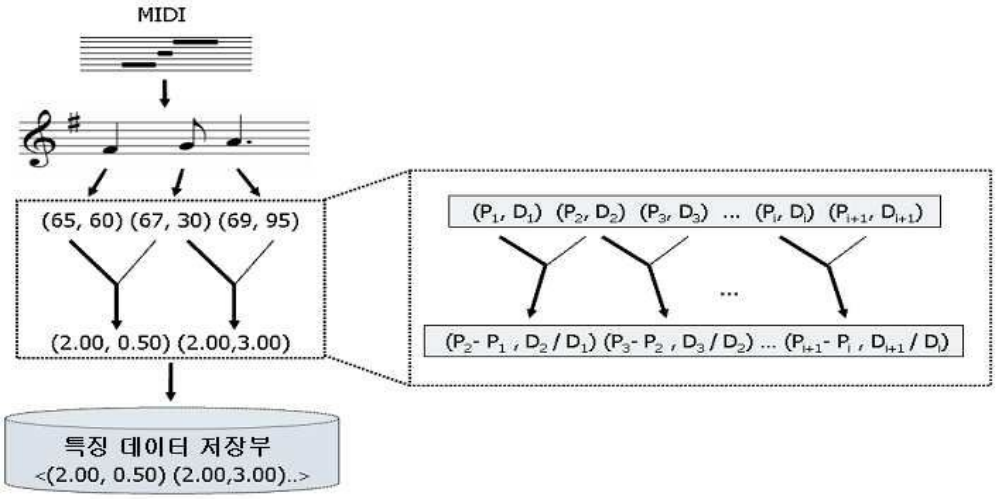
도면1


시 작
음악 파일 입력
110
멜로디를 나타내는 특징 데이터 추출
120
특징 데이터를 문자열로 변환
130
문자열을 제1 인덱스로 저장
140
변환된 문자열로부터 의미 있는 멜로디를
나타내는 문자열 추출하여 제2 인덱스로 저장
150
종 료

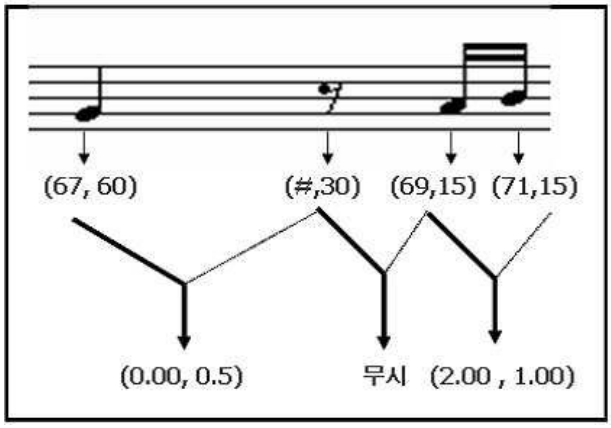
도면2


음악 데이터베이스
210
특징 데이터
저장부
220
제1 인덱스
230
제2-1 인덱스
240
제2-2 인덱스

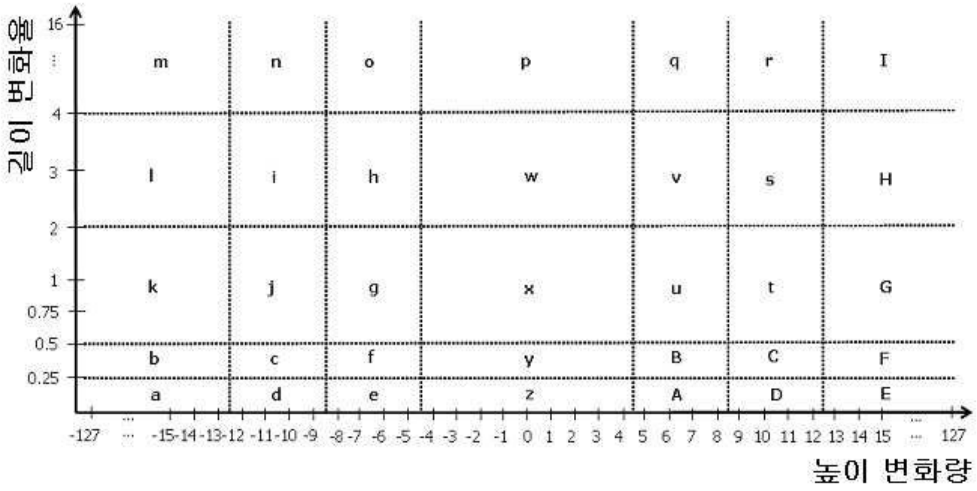
도면3


MIDI
(65, 60) (67, 30) (69, 95)
(2.00, 0.50) (2.00,3.00)
특징 데이터 저장부
<(2.00, 0.50) (2.00,3.00)..>
(P1, D1) (P2, D2) (P3, D3) ... (Pi, Di) (Pi+1, Di+1)
(P2- P1 , D2 / D1) (P3- P2 , D3 / D2) ... (Pi+1- Pi , Di+1 / Di)

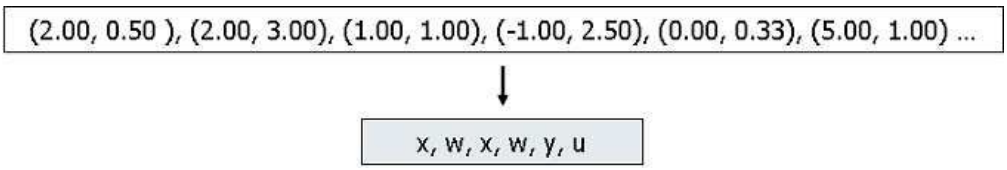
도면4


(67, 60)
(#,30)
(69,15)
(71,15)
(0.00, 0.5)
무시
(2.00 , 1.00)

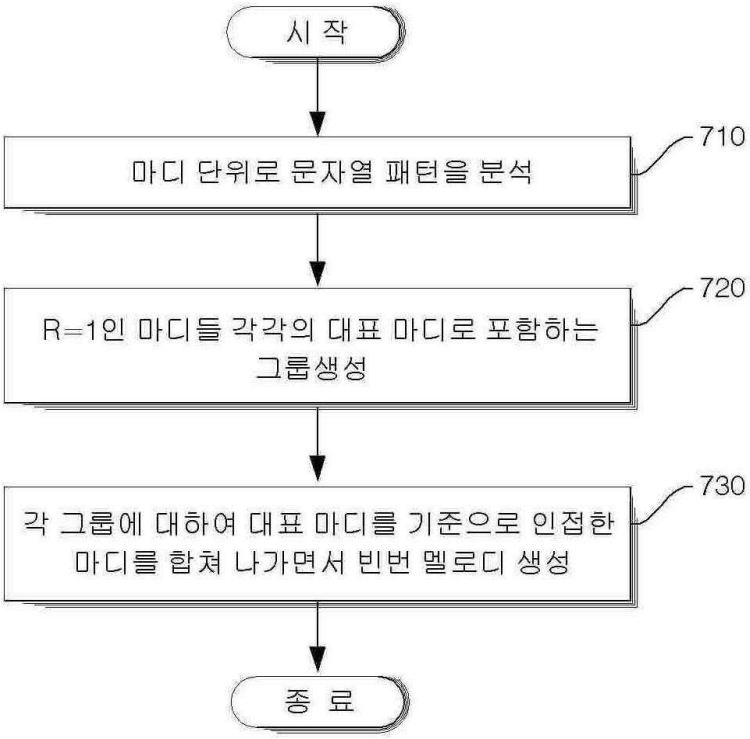
도면5


길이 변화율
16
4
3
2
1
0.75
0.5
0.25
m n o p q r I
l i h w v s H
k j g x u t G
b c f y B C F
a d e z A D E
-127 ... -15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 ... 127
높이 변화량

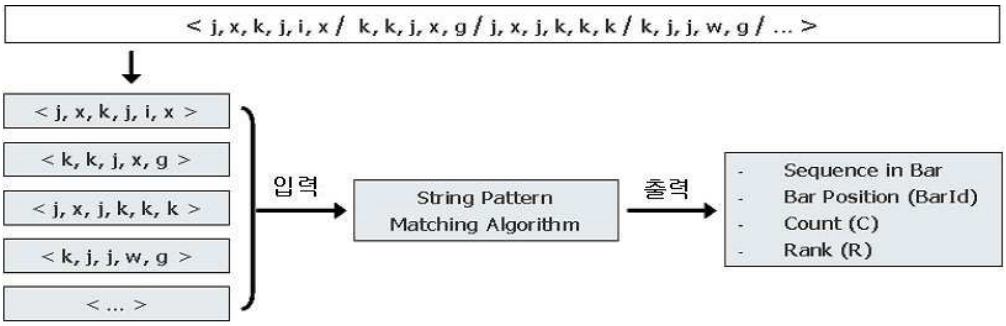
도면6


(2.00, 0.50), (2.00, 3.00), (1.00, 1.00), (-1.00, 2.50), (0.00, 0.33), (5.00, 1.00) ...
x, w, x, w, y, u

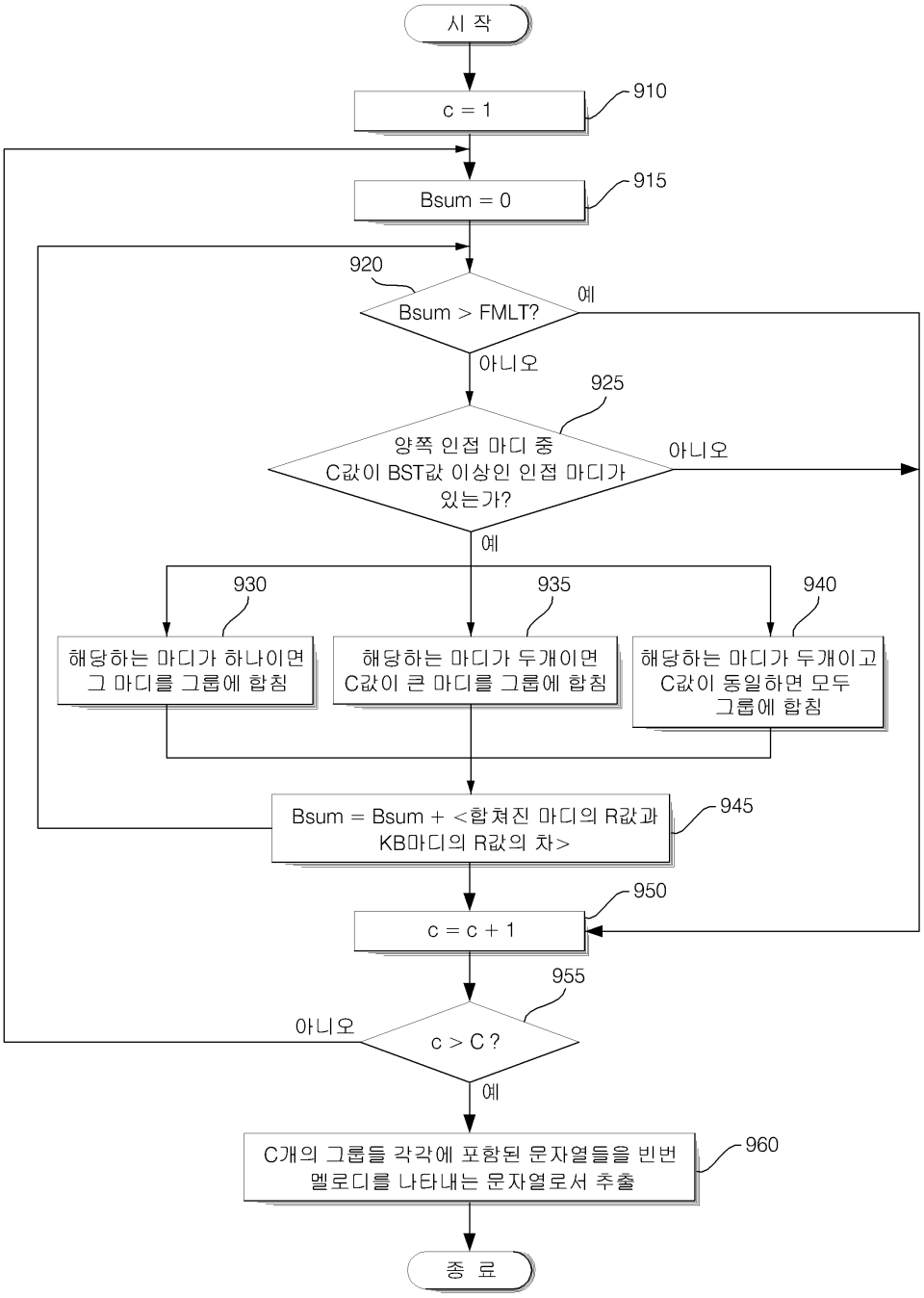
도면7


시 작
마디 단위로 문자열 패턴을 분석
710
R=1인 마디들 각각의 대표 마디로 포함하는 그룹생성
720
각 그룹에 대하여 대표 마디를 기준으로 인접한 마디를 합쳐 나가면서 빈번 멜로디 생성
730
종 료

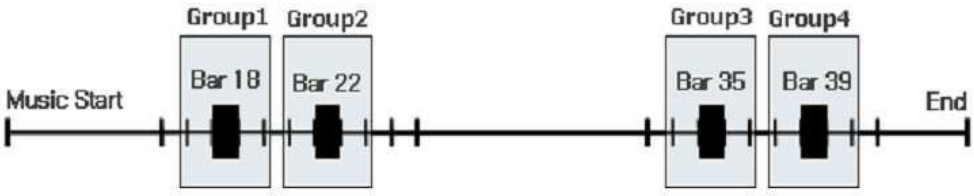
도면8


< j, x, k, j, i, x / k, k, j, x, g / j, x, j, k, k, k / k, j, j, w, g / ... >
< j, x, k, j, i, x >
< k, k, j, x, g >
< j, x, j, k, k, k >
< k, j, j, w, g >
< ... >
입력
String Pattern Matching Algorithm
출력
- Sequence in Bar
- Bar Position (BarId)
- Count (C)
- Rank (R)

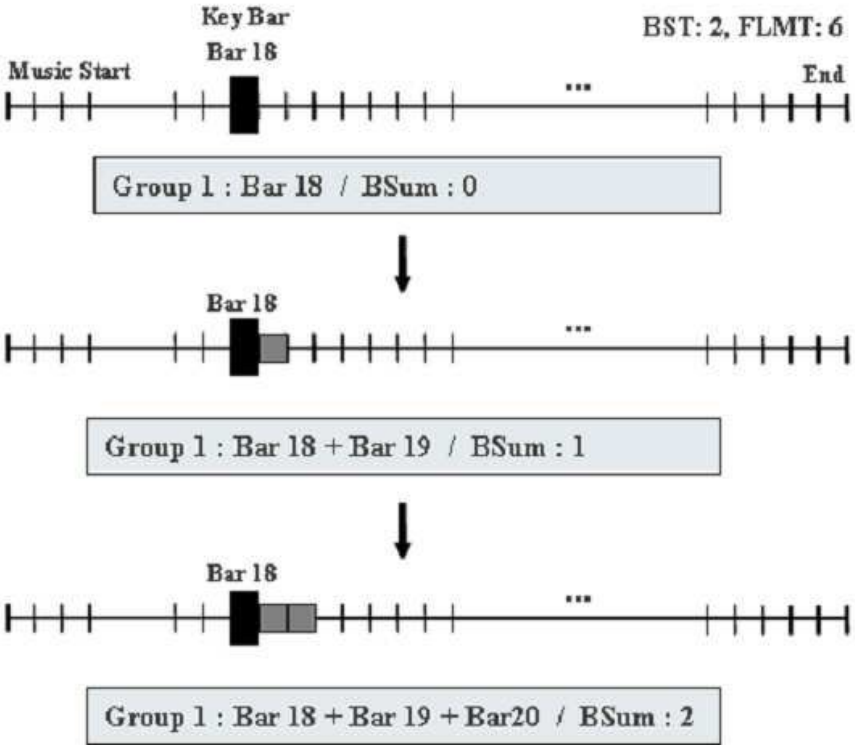
도면9


시 작
c = 1
910
Bsum = 0
915
920
Bsum > FMLT?
예
아니오
925
양쪽 인접 마디 중
C값이 BST값 이상인 인접 마디가
있는가?
아니오
예
930
935
940
해당하는 마디가 하나이면
그 마디를 그룹에 합침
해당하는 마디가 두개이면
C값이 큰 마디를 그룹에 합침
해당하는 마디가 두개이고
C값이 동일하면 모두
그룹에 합침
Bsum = Bsum + <합쳐진 마디의 R값과
KB마디의 R값의 차>
945
c = c + 1
950
955
아니오
c > C ?
예
C개의 그룹들 각각에 포함된 문자열들을 빈번
멜로디를 나타내는 문자열로서 추출
960
종 료

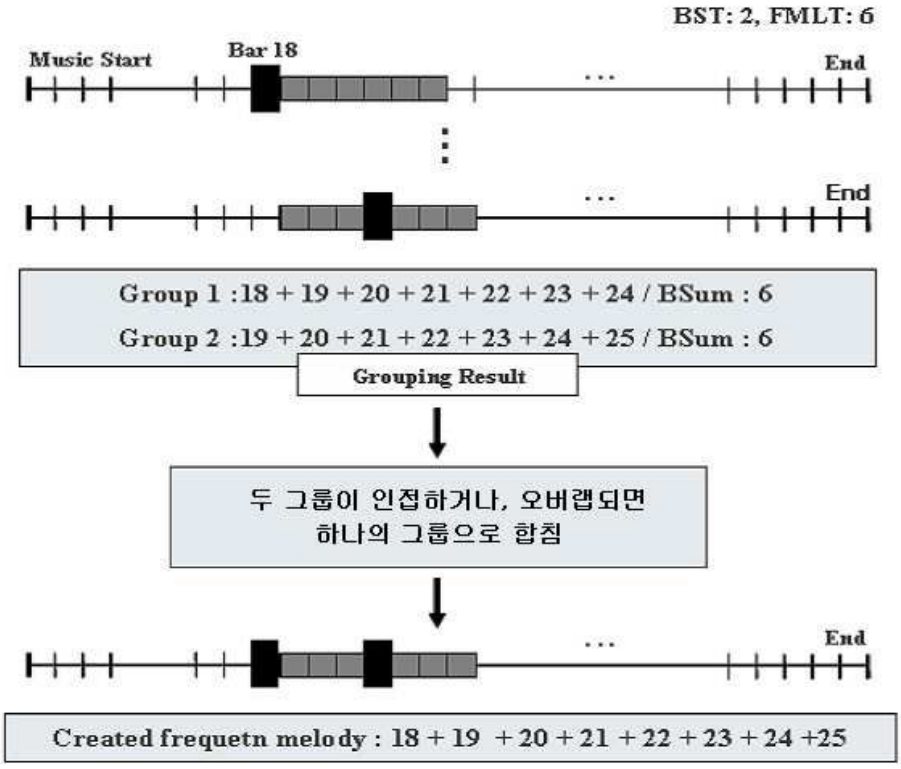
도면10


Group1
Group2
Group3
Group4
Music Start
Bar 18
Bar 22
Bar 35
Bar 39
End

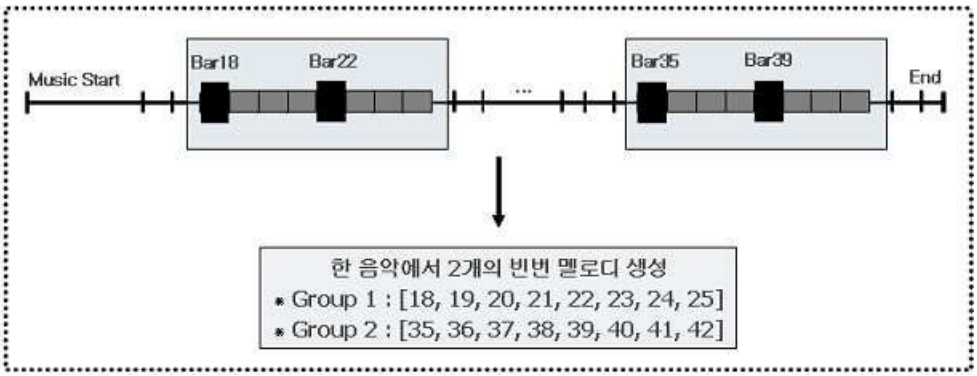
도면11


Key Bar
Bar 18
BST: 2, FLMT: 6
Music Start
End
Group 1 : Bar 18 / BSum : 0
Bar 18
Group 1 : Bar 18 + Bar 19 / BSum : 1
Bar 18
Group 1 : Bar 18 + Bar 19 + Bar20 / BSum : 2

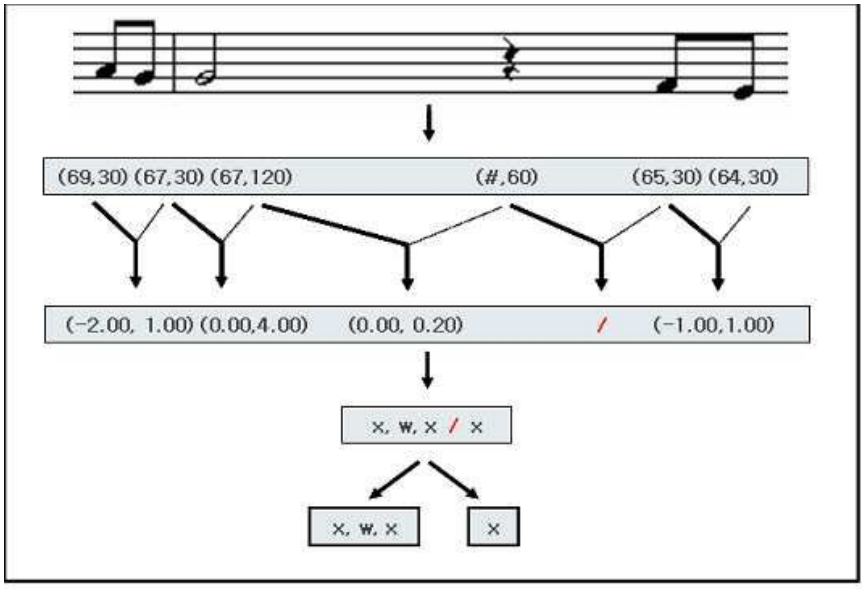
도면12


BST: 2, FMLT: 6
Music Start
Bar 18
End
End
Group 1 :18 + 19 + 20 + 21 + 22 + 23 + 24 / BSum : 6
Group 2 :19 + 20 + 21 + 22 + 23 + 24 + 25 / BSum : 6
Grouping Result
두 그룹이 인접하거나, 오버랩되면
하나의 그룹으로 합침
End
Created frequetn melody : 18 + 19 + 20 + 21 + 22 + 23 + 24 +25

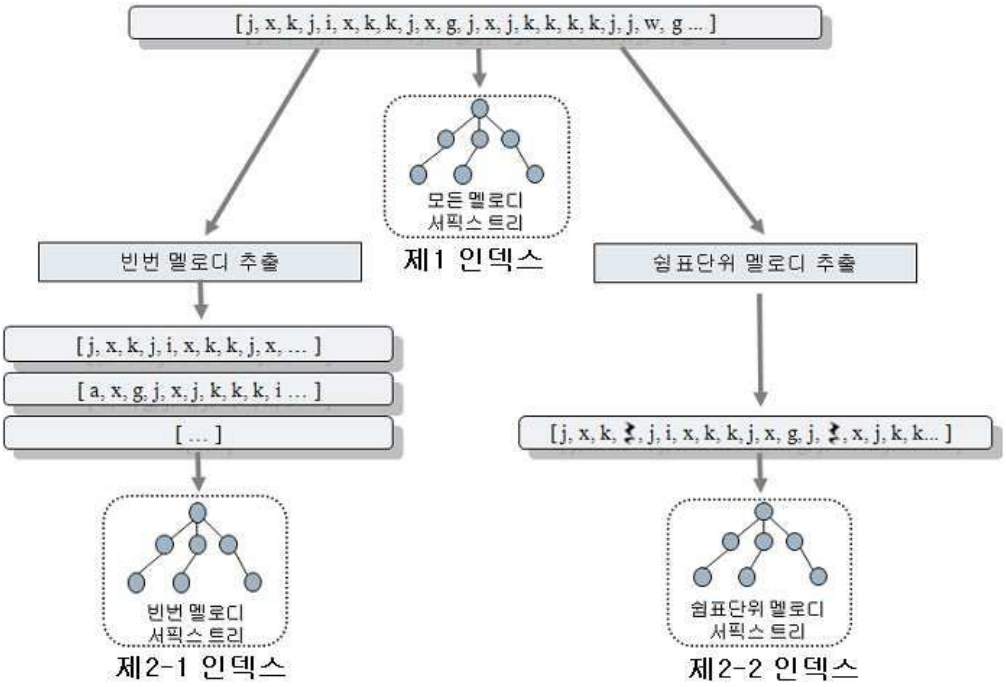
도면13


Music Start
Bar18
Bar22
...
Bar35
Bar39
End
한 음악에서 2개의 비번 멜로디 생성
* Group 1 : [18, 19, 20, 21, 22, 23, 24, 25]
* Group 2 : [35, 36, 37, 38, 39, 40, 41, 42]

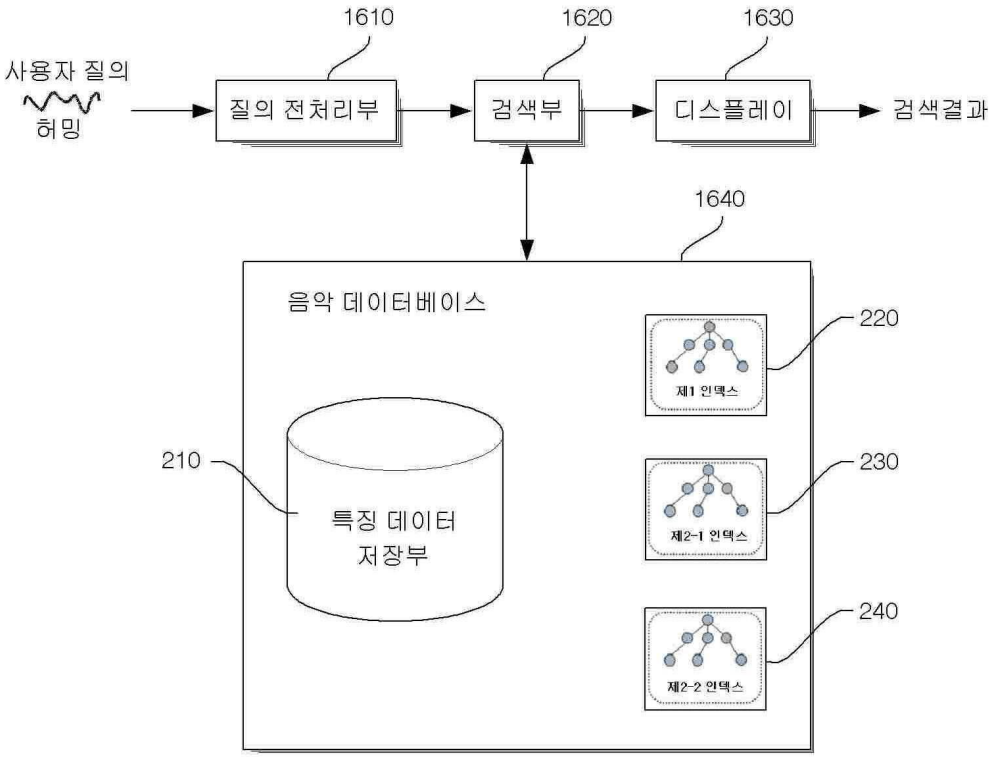
도면14


(69,30) (67,30) (67,120)
(#,60)
(65,30) (64,30)
(-2.00, 1.00) (0.00,4.00)
(0.00, 0.20)
/
(-1.00,1.00)
x, w, x / x
x, w, x
x

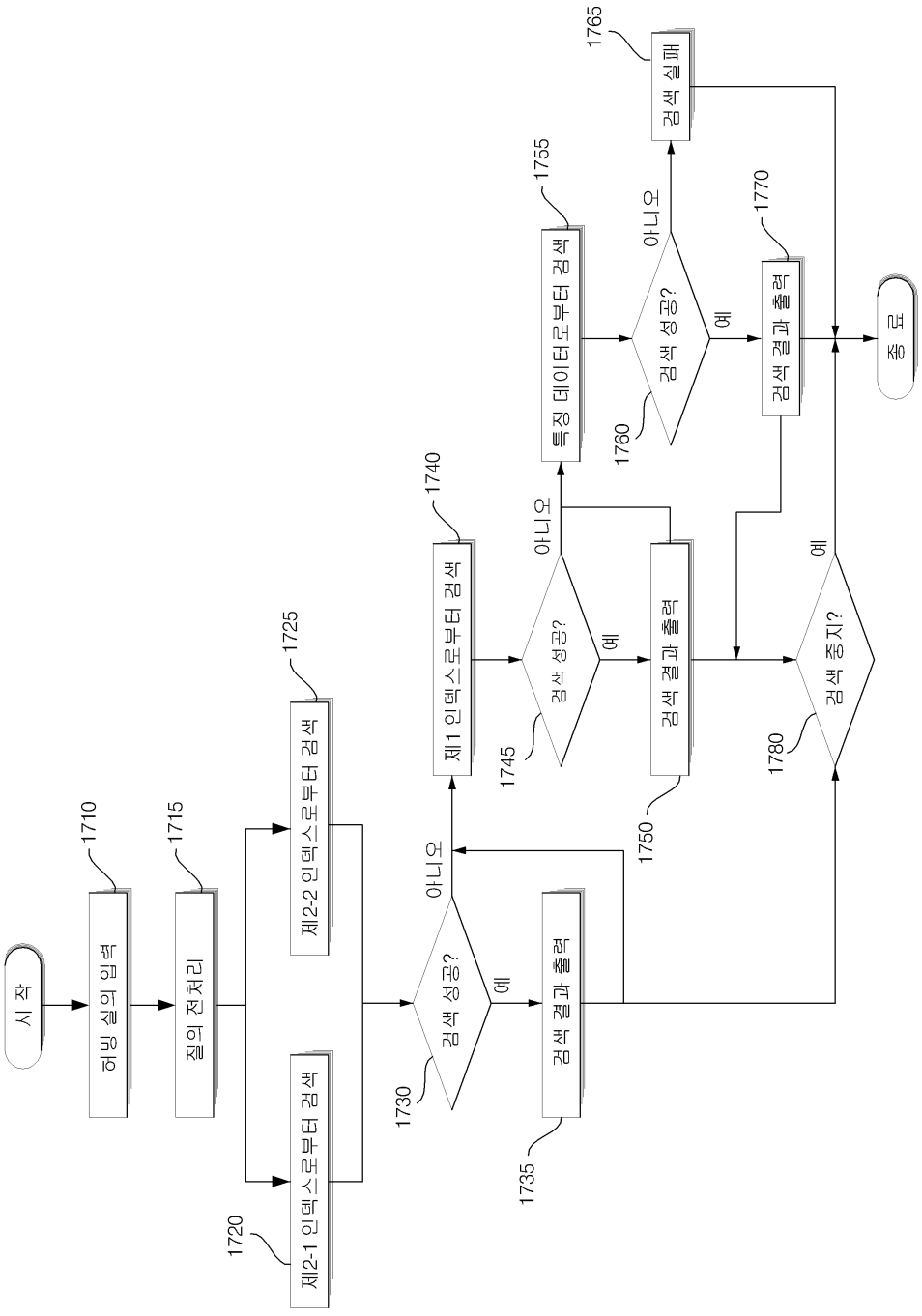
도면15


[j, x, k, j, i, x, k, k, j, x, g, j, x, j, k, k, k, k, j, j, w, g ...]
모든 멜로디 서픽스 트리
제1 인덱스
빈번 멜로디 추출
[j, x, k, j, i, x, k, k, j, x, ...]
[a, x, g, j, x, j, k, k, k, i ...]
[...]
빈번 멜로디 서픽스 트리
제2-1 인덱스
대표단위 멜로디 추출
[j, x, k, ♭, j, i, x, k, k, j, x, g, j, ♭, x, j, k, k...]
대표단위 멜로디 서픽스 트리
제2-2 인덱스

도면16


사용자 질의
허밍
1610
질의 전처리부
1620
검색부
1630
디스플레이
검색결과
1640
음악 데이터베이스
220
제1 인덱스
210
특징 데이터 저장부
230
제2-1 인덱스
240
제2-2 인덱스

도면17


시 작
허밍 질의 입력
1710
질의 전처리
1715
제2-1 인덱스로부터 검색
제2-2 인덱스로부터 검색
1720
1725
검색 성공?
1730
예
아니오
검색 결과 출력
1735
제1 인덱스로부터 검색
1740
검색 성공?
1745
예
아니오
검색 결과 출력
1750
특징 데이터로부터 검색
1755
검색 성공?
1760
예
아니오
검색 실패
1765
검색 결과 출력
1770
검색 중지?
1780
예
종 료