JDBC를 이용한 내용 기반 이미지 검색 시스템

Content-based Image Retrieval System Using JDBC

이상열, 안병규, 조세홍, 황병곤
Sang-Youl Lee, Byeung-kyui An, Se-hong Jo, Byeung-kyui An
대구대학교 정보통신공학부

초 록

본 논문은 웹상에서 이미지 검색 시스템을 구현하는데 검색 방법은 영상의 영역과 넓이를 이용한 체인 코드에 기반하여 복잡도와 영역 색상 정보를 이용하였고, 클라이언트와 서버간의 데이터베이스 연결은 JDBC를 이용하였다. 기존의 검색할 때마다 프로세스가 필요한 CGI를 이용한 방법보다 더 효율적이었다. 입력된 영상을 이용하여 검색하는 방법을 사용하였으며, 색상 정보 추출은 RGB신호를 256칼라로 양자화 하였다. 영상의 색상과 객체가 갖는 복잡도를 이용한 내용기반 영상 검색 방법을 제시하였다. 본 논문에서는 기존의 방법인 색상특징과 제안한 체인코드에 의한 객체의 복잡도를 특징으로 하는 공간정보를 결합한 방법을 제안하였다. 실험결과 영상의 모양 특징도 고려한 제안한 방법이 내용기반 검색에서 색상 특징만을 고려한 기존의 방법보다 우수하였다.

1. 서 론

최근 인터넷의 발달로 광범위하게 분산된 다양한 형식의 데이터를 손쉽게 검색할 수 있는 시스템이 개발되고 있다. 처음에는 텍스트 기반에 맞추어 HTTP, HTML, URL 등이 제안되었고 이를 통해 비 동기적인 형태의 검색과 단순하고 단일한 방식의 표현 방식이 사용되어 왔다. 그러나 최근 인터넷상에 상당수의 데이터들이 보다 복잡해지고, 구조화 되어가고 있으며, 동기적인 멀티미디어 정보를 포함하는 등 새로운 구조 및 표현 방식을 요구하게 되었다. 특히, 일반 텍스트 자료를 검색하는 경우보다 멀티미디어 정보를 검색하는 경우는 영상 정보를 이용하기 때문에 검색 구조부터 다르다. 영상자료를 검색하는 방법은 크게 다음의 두 가지로 분류될 수 있다.

첫째, 검색할 모든 데이터를 검색 데이터베이스에 저장할 때 운영자가 일일이 그림에 대한 주석을 기술하고, 검색자가 찾고자하는 그림의 주제를 문자로 검색하는 문자기반(Text-based) 검색 방법이다. 이 방법은 제한된 범위 내에서 의미 정보에 따른 영상 검색이 가능하나, 대용량의 영상 데이터의 경우 운영자가 일일이 주석을 기술해야 하고 사용자와의 관점의 불일치로 인하여 검색의 비효율성이 제기될 수 있다는 단점이 있다.

둘째는 문자기반 검색 방법의 단점을 극복하기 위해 영상 데이터에서 표현되는 특징(Feature)들을 자동으로 추출하여 검색하는 내용기반(Content-based) 검색 방법이 있다 [1,2]. 이 방법은 주석과 관계없이 칼라(Color), 모양(Shape), 질감(Texture) 등을 영상의 내용 표현 요소들을 통하여 얻어진 영상들간의 유사도를 계산하고 이에 따라 영상 검색을 수행한다. 그러므로, 효율적인 영상 검색을 위해서는 영상의 내용 표현요소들에 대한 효과적인 특징 추출이 무엇보다도 중요하다.

위의 검색 방법을 웹에서 구현하려면 클라

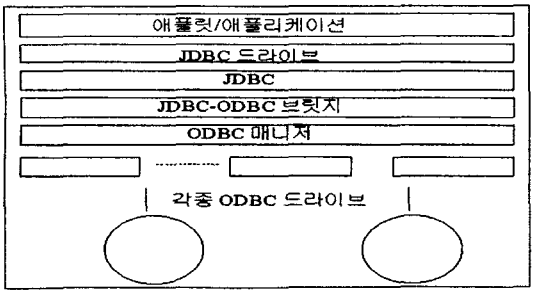
이언트의 영상 정보를 분석하고, 분석된 자료를 서버의 데이터 베이스와 연결하여 정보를 주고 받을 수 있어야 한다. 그러나, 이러한 정보를 지원하고 웹브라우저가 없기 때문에 클라이언트와 서버간의 데이터베이스 접속이 어렵다. 본 논문은 어러한 문제를 해결하기 위해서 JDBC를 사용하여 검색 시스템을 구현한다.

본 논문의 전체적인 구성은 2장은 본 논문에서 제안하고 있는 웹에서 이미지 검색 시스템을 구현하는데 데이터 베이스를 JDBC를 이용하여 구축하는 방법과 구조에 대해 설명하고, 3장은 내용기반 영상 검색 시스템의 구조에 대하여 살펴본다. 그리고, 4장은 실험을 위한 구현 환경 및 추출된 다양한 특징 값을 이용하여 각 방법들간의 실제적인 검색 효율을 비교·분석한다. 마지막으로, 5장은 본 논문의 결론을 맺기로 한다.

2. JDBC 기반 이미지 검색 시스템

1) 데이터 베이스 연결 종류

웹상에서 클라이언트와 서버간에 데이터베이스로 연결하는 것은 여러 가지 방법들이 있다. SQL문을 HTML에 내장시켜서 사용하는 확장 HTML 방법, 데이터베이스에 저장된 테이블과 필드들에 관한 정보를 외부 템플릿 파일에 저장한 다음, 사용자가 요청하면 이 정보를 이용하여 동적 SQL 질의를 생성하는 템플릿 파일 방법, 일련의 SQL 명령어들이 데이터 베이스에 저장한 다음 필요할 때 외부 프로그램에서 호출할 수 있는 데이터베이스 저장 프로시저 기능을 이용하는 데이터 베이스 저장 프로시저 방법, 동일한 데이터베이스 응용 프로그램으로 다양한 데이터베이스를 접근할 수 있도록 산업계 표준 인터페이스인 ODBC (Open DataBase Connectivity)를 이용하는 표준 인터페이스 방법, 특정 데이터베이스엔진이 제공하는 고유의 API를 이용하는 고유 API 방법, 자바 서블릿(servlets) 접근 방법 등이 있다. 클라이언트와 웹서버간에 HTTP 메시지 처리 과정에서 하나의 요구(request)와 하나의 응답(response)이 서로 독립적으로 수행하는 CGI는 사용자가 HTML FORM 필드의 내용을 전송할 때마다 매번 CGI 프로그램을 다시 실행하게 된다. 즉, CGI를 이용한 데이터 베이스와의 연동은 단 방향성 프로토콜이기 때문에 사용자가 질의할 때마다 매번 데이터베이스와 연결하고 해제하는 프로세스가 수행하게 된다. 그러나, JAVA 기반의 연동 방법은 서블릿(servlets)을 이용하는 방법이 사용하므로 CGI 접근 방법에서 발생하는 문제를 해결 할 수 있다. 서블릿은 서브쪽에서 실행되는 애플릿(applet)으로 자바 서버 API의 서브셋이다. 서블릿은 CGI와 유사하게 HTML 폼(form)이나 클라이언트 애플릿을 통해서 PUT/POST 메소드로 사용자 질의를 전달 받는다. 질의를 전달 받은 서블릿은 JDBC를 통해서 데이터베이스 서버에 연결하고, 실행결과를 반환 받아서 HTML 문서로 포맷팅 한 후 클라인트로 전송한다. <그림 1>은 이러한 연동구조를 나타난 것이다. 서블릿은 자바 멀티 쓰레드를 이용해 실행되기 때문에 CGI 방식처럼 다수의 프로세스 생성으로 인한 성능 저하 현상이 발생하지 않는다. 서블릿을 이용한 자바기반 접근방법은 상태 유지가 가능하고 JDBC를 통한 데이터 베이스 연결이 쉽다는 장점을 가진다. JDBC는 Sun Microsystem사에서 개발한 SQL 데이터 베이스 접근 인터페이스로서, JDBC API는 데이터 베이스 연결, SQL 명령문, 질의 결과 집합, 메타 데이터 등을 표현 할 수 있다.



<그림 1> JDBC 연동구조

2) JDBC 접속

JDK(JAVA Development Kit) 에는 JDBC를 위한 java.sql.* 클래스들이 있다. JDBC 2.0 표준 외에 근래에 몇 가지 기능이 추가된 Standard Extension API를 담고 있는 javax.sql.* 클래스들이 있는데 이 클래스는 JDBC API의 인터페이스 정의에 대한 것만 담고 있을 뿐, 개별 데이터베이스와는 무관하다. 따라서, Java가 제공하는 JDBC API를 사용하기 위해서는 개별 데이터베이스에 맞게 구현된 드라이버가 있어야만 한다.

Connection 객체는 데이터베이스와의 연결(connection)을 담당한다. 연결 부분은 실행될 SQL문, 그리고 연결을 통해서 반환되어지는 결과들을 포함한다. 어플리케이션은 한 개의 데이터베이스와 하나 이상의 연결을 할 수 있거나 또는 많은 서로 다른 데이터베이스에 연결할 수도 있다. 다음은 JDBC를 이용하여 데이터베이스를 연결하는 프로그램이다.

```
import java.sql.*;
import java.lang.*;
public class Test {
  public static void main(String[] args) {
     // 드라이버 로딩
    try {
      Class.forName("myDriver.ClassName");
      } catch (ClassNotFoundException e) {
        System.err.println("Class Not Found : " +
e.getMessage());
       }
      // 데이터베이스 접속
       try {
        String url = "jdbc:myprotocol:mydatabase";
        Connection db =
          DriverManager.getConnection(url, "myid",
            "mypassword");
        } catch (SQLException e) {
         System.err.println("SQL Error : " +
              e.getMessage());
       }
    }
}
```

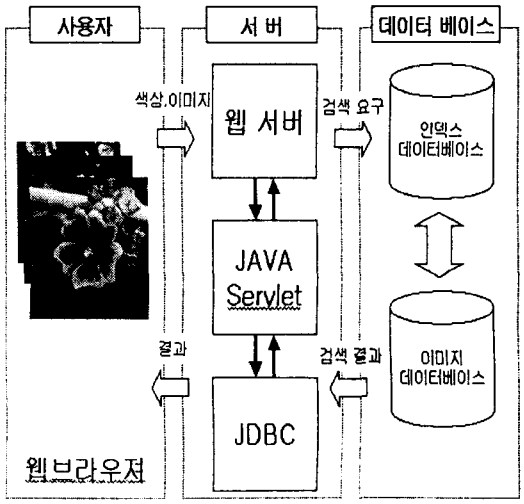
<그림2> JDBC 데이터베이스 연결

3. 내용기반 영상 검색 시스템

만일 사용자가 웹 상에 존재하는 많은 이미지 중 자신이 원하는 이미지를 보고 싶다고 하자. 사용자가 원하는 이미지에 대해 적절히 설명하면 검색 엔진이 이와 가까운 대상부터 보여준다. 사용자는 이중에서 자신의 의도에 맞는 것을 고를 수 있다. 이러한 검색 엔진이 사용자의 질의 요청이 들어온 다음부터 웹사이트들을 하나하나 검색할 수는 없으므로, 사용자의 질의 요청이 없을 때에도 미리 웹사이트에 있는 문서를 돌아다니면서 이미지와 관련 있는 텍스트와 이미지의 색상 히스토그램을 가져온다. 사용자가 원하는 이미지를 설명하는 방법은 이미지 설명 텍스트와, 이미지 특성 중 색상 히스토그램과 체인 코드를 사용한다. 질의 방법은 다음과 같다.

첫째, 검색 엔진에서 제공하는 색상 히스토그램 표를 질의자가 선택하여 검색할 수 있도록 했다.

둘째, 사용자가 특정 이미지를 입력하면 이미지를 분석하여 검색 엔진에 색상 히스토그램과 체인코드 값을 입력한다. 입력받은 검색 엔진은 데이터베이스에 저장되어 있는 이미지의 색상 히스토그램과 체인 코드를 비교하여 사용자가 입력한 이미지와 유사한 이미지를 출력한다. 이미지 특성만 이용하거나 이미지 설명 텍스트 또는 이미지의 이름을 입력하여 검색할 수 있다. <그림2>는 JDBC를 이용한 이미지 검색 시스템의 전체 구조를 나타내고 있다.



<그림2> 자바 기반의 JDBC 연동구조

1) 색상 정보를 이용한 검출

색상 정보를 이용하기 전에 이미지 검색을 위한 두 단계를 거치게 된다. 첫째, 검색하는 이미지가 각각 크기가 일정하

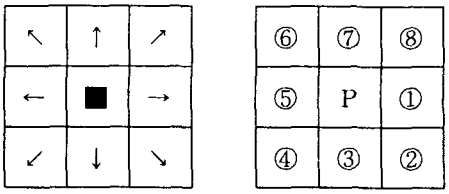
지 않으므로 영상의 크기를 320X240으로 일정하게 하여 같은 크기로 비교할 수 있도록 한다. 둘째, 24bit칼라영상은 많은 양을 계산하므로 처리속도가 저하되어 검색하는데 비효율적이므로 색상 비교하는데 크게 영향을 미치지 않는 범위인 8bit의 256칼라로 양자화 한다. 이런 과정을 마친 이미지는 정보를 표현하기 위하여 Swain이 제안한 색상 히스토그램[3,4]을 많이 사용한다. 장점으로는 이미지 성질을 대표할 수 있고 알고리즘이 간단하며, 물체의 회전이나 작은 이동 등과 같은 기하학적인 변형에 잘 활용할 수 있다. 그러나 빛의 밝기와 영상내의 물체의 크기에 민감하고, 전혀 다른 영상도 같은 색상 분포를 갖는 단점이 있다. 칼라 특징 정보는 R, G, B로 표현되는 칼라 이미지의 최대 히스토그램을 좌표로 표현할 수 있다. 각 색상 값에 대해 히스토그램은 (1)과 같이 생성된다.

$$H = Max(n(i)) \qquad (1)$$

식(1)에서 이미지의 전체 픽셀수가 n, 특정 칼라값이 i, i 칼라 값을 갖는 픽셀의 합을 $n(i)$로 두었다. 각 칼라값에 대한 히스토그램의 최대값 H를 이미지의 키 값으로 사용하였다.

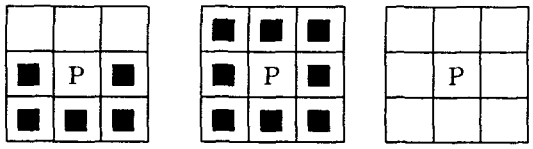
2) 체인 코드를 이용한 복잡도 계산

사용자가 질의한 영상의 특징을 추출하기 위해서 체인코드를 이용하였다. 체인코드를 이용하기 위해서는 영상을 이진화 한다. 이진화 된 영상에 체인 코드를 이용하여 객체의 외곽선 길이와 넓이를 구하게 된다. 이들 값을 이용하여 복잡도를 산출한다. 체인 코드는 점의 특성을 파악하여 노이즈를 제거함으로 더 정확한 복잡도를 구할 수 있다. 체인코드를 구하는 방법은 처음 왼쪽 상단부터 시작하여 객체가 발견될 때까지 우측 하단으로 추적한다. 만약 객체를 발견하면 <그림3>와 같은 순서로 체인코드를 찾기 시작한다.



<그림3> 8방향 체인코드

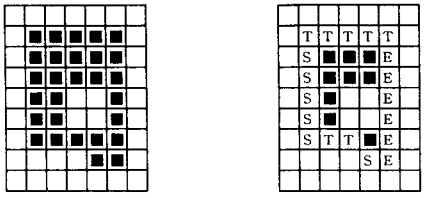
체인코드를 이용하여 외곽선을 찾아 영역의 수를 추출하는 방법으로 시작점 P에서 8방향의 순서로 다음 점을 추적한다.



P = 외곽 점 P = 내부 점 P = 고립 점

<그림4> 점의 위치에 따른 특성

<그림4>과 같이 추적 후 8방향 중 한 방향에 한 개의 점이 추출되면 외곽선을 이루기 위한 시작점이 되고 다음 점으로 P가 옮겨진다. 외곽선 추적은 시작점을 다시 만날 때까지 계속되며 시작점을 만나게 되면 한 개의객체로 인식한다. 그러나 시작점 P에서 다음 점을 추적할 수 없을 때는 고립점으로 판단하여 디지털 신호에서 올 수 있는 잡음으로 판단하여 계산과정에서 제외시킨다. 점 P를 발견한 후 고립점이 아닐 경우 윤곽선을 시계 방향으로 왼쪽화소 우선 추적법으로 추적한다. 추적할 때 출발점에서 옆으로 진행이 되면 <그림5>와 같이 T기호를 부여하고 다음 점이 아래로 향하면 출발점에 T를 부여하고 도착점에 E를 부여하고 다음 점이 위로 향하면 출발점에 S를 부여한다. 추적 시작점으로 되돌아오면 왼쪽 아래 화소 연결을 확인하여 영역의 길이 값을 저장한다 그리고 S에서 E사이를 T로 채우고 T의 개수가 영역의 넓이 값으로 저장한다. 만약 내부 영역을 발견하면 시계 반대 방향으로 다시 추적을 하면서 S, E, T를 부여한다. 모든 영역이 T로 바뀔 때까지 반복하여 수행한다.



a. 입력된 2진 영상 b. 1차 추적

c. 2차 추적 d. 3차 추적

<그림5> 윤곽선추적 알고리즘 적용 예

체인코드를 이용하여 객체의 면적과 외곽선의 길이를 구하여 복잡도를 얻는다. 복잡도 Co는 식(2)과 같다. 여기서 L은 체인코드에서 구한 영상의 외곽선 길이를 나타내며 M은 영상의 영역의 넓이를 나타낸다.

$$Co = L^2/M \qquad (2)$$

3) 데이터 베이스 구조 및 유사도 측정

(1) 데이터 베이스 구조

영상의 이미지는 데이터 베이스에 포함하지 않고 파일 정보만 가지고 있어 영상을 쉽게 수정할 수 있도록 하였다. 대표 색상값 및 복잡도는 영상 자료를 각각 읽어 데이터 베이스에 추가할때 자동적으로 만들어진다. 질의를 수행할 경우 인덱스키를 사용하여 검색한 후, 영상의 특징값 들을 읽어와 사용자가 질의한 영상과 서로 비교한다. 해당 인덱스키의 비교시 질의한 키와 참조키가 정확히 일치되는 영상뿐만 아니라 질의한 값과 유사도가 큰 순서부터 작은 순서로 검색하여 나타낸다.

image ID(이미지 ID)	char	5
name(등록자)	char	10
subject(주제)	char	30
visited(검색횟수)	integer	
date(등록일)	date	
content(설명정보)	char	30
chaincode(체인코드값)	integer	

(a) 인덱스 데이터베이스

image ID(이미지 ID)	char	5
Rcolor(Red)	integer	
Gcolor(Green)	integer	
Bcolor(Blue)	integer	

(b) 색상정보 데이터베이스

image ID(이미지 ID)	char	5
isource(이미지 소스)	BLOB	

(c) 이미지 데이터베이스

<표1> 데이터 베이스 필드

(2) 유사도 측정

유사도란 질의 영상과 데이터베이스에 저장된 영상들의 유사성 측정을 하는 것이다. 유사도를 구하기 위한 데이터베이스의 필더는 전체 색상 대표값 DL_i, 내부영역 대표값 DS_i, 복잡도 DC_i라고 둔다. 여기서 i는 데이터베이스의 레코드 번호이다. 질의 영상이 입력되면 <그림1>의 처리과정을 거쳐 질의 영상에 대한 전체 색상 대표값을 QL, 내부영역 대표값을 QS, 복잡도를 QC라고 한다. 각각의 데이터를 이용한 유사도 R_i는 식(3)으로 나타낸다.

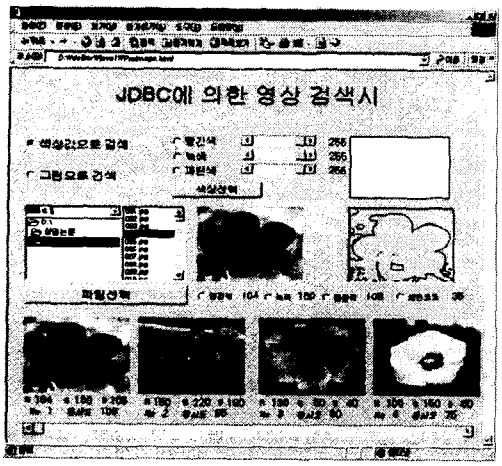
$$R_i = \left(1 - \frac{\sum_i (|DL_i - QL| + |DS_i - QS| + |DC_i - QC|)}{\sum_i ((DL_i + QL) + (DS_i + QS) + (DC_i + QC))}\right) X100 \qquad (3)$$

본 실험의 경우 데이터 베이스에 100개의 영상 자료를 가지고 실험하였다. <식3>에 의해 얻어진 실험결과를 내림차순으로 재 정렬하여 유사도가 100에 가까운 것부터 4개씩 나타내었다.

4. 실험 및 평가

1) 실험 환경

본 논문에서 제안된 시스템의 구현 환경은 한글 윈도우즈 98을 운영체제로 하는 펜티움 400MHz 상에서 실험하였다. 실험 영상은 다양한 색과 영역의 부분 또는 전체를 차지하여 영역별 변화가 많은 꽃 영상을 사용하였다. 꽃 영상의 경우 내부영역을 차지하는 경우 전체 영역으로 퍼진 경우 등 다양한 데이터를 볼 수 있다.



<그림4> 이미지 검색 결과

.2) 체인코드를 이용한 복잡도 계산

영상의 색상 정보만을 이용하면 같은 서로 비스한 빨간색의 경우 구별하기 힘든 경우가 있다. 이런 단점을 보안하기 위해 복잡도를 이용하여 영상을 구별할 수 있다. <그림9>는 대표 색상값을 추출하기 위한 작업과 그 결과 값을 기준으로 한 복잡도 추출을 위한 작업을 병행함으로 다양한 경우의 수를 만들었다.

<그림5> 체인코드를 이용한 공간값 계산

5. 결론

지금까지 웹 상의 이미지를 내용 기반 검색하기 위한 WWW에 의한 내용 기반의 영상 정보 검색에 대해 기술하였다.

현재 영상 검색 시스템들은 대부분이 텍스트에 의한 검색만을 지원하고 있는 실정이다. 이는 사용자들이 그 동안 텍스트를 이용한 검색 방법에 익숙해져 있기 때문으로 볼 수 있지만 아직 내용 기반에 의한 검색 기술이 실용적으로 쓰일 만큼 높은 성능을 보이고 있지 않기 때문이다. 따라서 좀더 효율적이고 정확한 검색 기술의 개발이 필수 적인데 이를 위해서는 영상에서의 특징 추출과 다차원 특징 정보그이 색인화 기술, 데이터베이스 연동 기술이 서로 유기적으로 연구되어야 한다.

참 고 문 헌

[1] Batber, W. Equitz, C. Faloutsos, "Query By Content for Large On-Line Image Collection", IEEE, 1995.

[2] Y. H. Ang, Zhao Li and S. H. Ong, "Image Retrieval based on Multimensional Feature Properties", SPIE Vol. 2420, pp.47~57, 1995.

[3] 김진아, 정성환, "내용기반 영상 데이터 검색을 위한 질감의 통계적 기법", 한국정보과학회 영남 지부 '97학술 표논문집, 제4권, 제1호, pp.85~88, 1997.

[4] 김회승, "영상인식-영상처리, 컴퓨터비젼, 패턴인식, 신경망", 생능출판사, 1994.

[5] Ramesh Jain. Rangachar Kasturi. Brian G, Schunck, "Machine Vision", ISBN 0-07-032018-7, pp.234~248, 1995.

[6] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, W. E quitz, "Efficient and Effective Querying by Image Content," Journal of Intellige nt Information Systems, 3, pp. 231-262, 1994.

[7] Ramesh Jain, Rangachar Kasturi, Brian G, Schunck, "Machine Vision", ISBN 0-07-032018, pp.234-248, 1995.

[8] Ramesh Jain, S.N. Jayaram Murthy, Pe ter L-J Chen, "Similarity Measures for Image Database", SPIE Vol.2420, 1995.

[9] Ramesh Jain. S.N. Jayaram Murthy. Peter L-J Chen, "Similarity Measures for Image Database", SPIE Vol. 2420, 1995.