Lab 1. Digital Image 의 이해

실습 목표: 디지털 이미지의 종류와 구성에 관하여 살펴봄으로써 디지털 이미지에 관한 이해를 높이고, 디지털 이미지의 컬러 값 변환에 관한 실습을 통하여 다양한 이미지 처리 프로그램의 내부에서 수행되는 기본적인 연산을 이해한다.

목차

- 디지털 이미지
- 비트맵 (Bitmap) 이미지의 구조
- 16진수를 이용한 비트 연산
- 컬러 변환 프로그램 구현
- 숙제: 효율적 컬러 변환, OpenCV를 이용한 컬러 변환
- 참고문헌

디지털 이미지

디지털 이미지는 크게 레스터(Raster) 방식과 벡터(Vector) 방식으로 나뉜다. 레스터 방식은 일정 수의 열(column)과 행(row)으로 이루어져 있으며 특정 열과 행 값은 픽셀(Pixel)이라 부르고, 각 픽셀은 컬러나 밝기를 나타내는 정량화 된 값을 가진다. 각 픽셀 값, 또는 컬러 이미지의 경우 각 픽셀의 컬러 값은 8비트로 표현되며, 더 정교한 표현을 위해 16 비트 값을 사용하기도 한다. 한편, 벡터 방식은 선, 커브, 도형 등의 기본 모양을 포맷화된 수식의 형태로 저장한다. 따라서 레스터 방식의 이미지는 세밀한 부분까지 자유롭게 묘사가 가능한 반면, 이미지의 크기가 변함에 따라 원래의 선명함을 유지하기가 어려우며, 벡터 방식은 표현할 수 있는 형상에 제약이 있지만 크기 변환에도 선명도를 유지할 수 있다. 레스터 이미지로는 BMP, PNG, TIFF, JPEG, GIF 등이 있으며, 벡터 이미지로는 EPS, SVG, PDF 등이 있다.

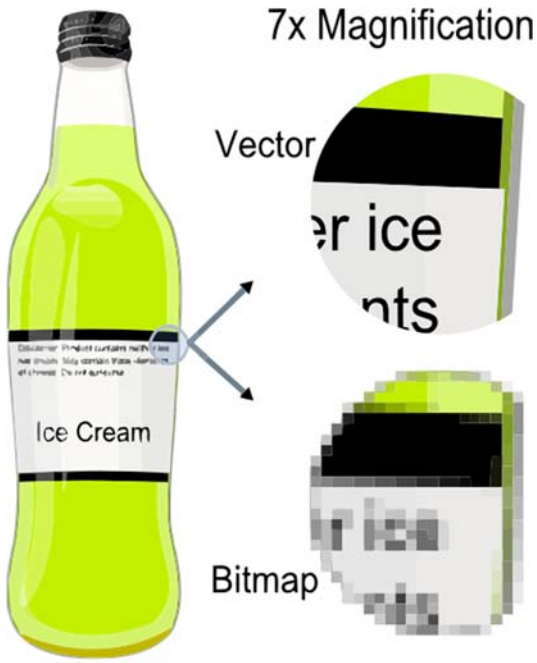

그림 1. Bitmap 이미지와 벡터 이미지의 차이. (http://en.wikipedia.org/wiki/Vector_graphics)

또한, 레스터 이미지는 비압축형과 압축형으로 나뉘며, BMP, PNG, TIFF 등이 대표적인 비압축형이고, JPEG, GIF 등이 대표적인 압축형 이미지이다. 압축형으로 가장 많이 쓰이는 JPEG (Joint Photographic Experts Group) 는 1994년 ISO(International Standardization Organization, 국제 표준화 기구)의 인증을 받으며 디지털 이미지 규격의 세계 표준으로 자리잡게 된다. JPEG 이미지는 '제이펙'이라고 읽어야 하지만, jpg 확장자가 많이 쓰이면서 '제이피지'라고 불리기도 한다. JPEG는 또한 손실형 압축 이미지이며, 비손실압축 JPEG 규격도 있으나, 용량이나 특허 문제로 인해 현재 거의 쓰이지 않는다. JPEG는 화질대비 압축률이 GIF에 비해 월등히 높으며, 최대 16,777,215색

(24비트 컬러)까지 표현이 가능하며 사용자가 압축률을 조정하여, 화질/용량의 밸런스를 조절할 수 있다. GIF 는 비손실 압축 이미지이며 256 가지 색만이 표현 가능하고, 특허로 보호 되고 있다. 비트맵 이미지도 압축 규격이 정의되어 있기는 하지만 보통 비압축형으로 많이 사용된다. 일반적으로 많이 사용되는 방식은 레스터 방식이므로, 실습에서는 주로 레스터 이미지, 특히 BMP(비트맵) 이미지에 관하여 논의하게 된다.

컬러 이미지는 보통 R, G, B 의 세가지 구성 요소를 가지며 이를 컬러 채널(color channel)이라고 한다. 그림 2 는 컬러 이미지의 세가지 컬러 채널과, 각각의 컬러 채널을 grayscale로 표현했을 때의 이미지를 보여준다.

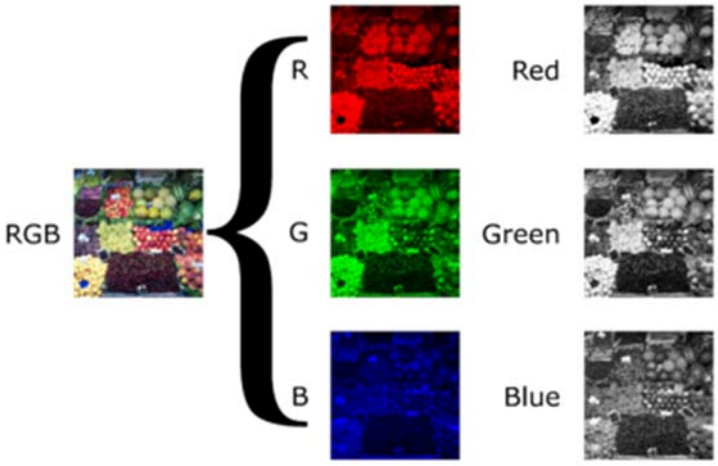

그림 2. 컬러 이미지의 R, G, B 채널 (http://en.wikipedia.org/wiki/Grayscale)

비트맵 (Bitmap) 이미지의 구조

비트맵 이미지 파일은 헤더, 인포헤더, 팔렛, 이미지 데이터로 구성된다. 헤더는 이미지의 기본 정보를 제공한다. 헤더에서 가장 중요한 필드는 offset 으로, 실제 이미지 값들이 시작하는 지점을 나타낸다. 인포헤더는 40 bytes 길이로, 이미지의 주요 속성 정보를 제공한다. 24 비트 이미지(트루 컬러)의 경우, 컬러 팔레트는 생략되고 이미지 데이터가 인포헤더의 바로 다음에 위치한다. 이 경우 이미지 데이터의 각 바이트는 파랑, 초록, 빨강 색을 나타내고 0 는 검정, 255 는 최대 컬러 값을 갖는다. 24 비트 이미지 이외는 Indexed 이미지라고 하며, 팔레트에서 픽셀 값이 색상표에 어떻게 매치되는지를 정의해 주어야 한다.

비트맵 이미지는 또한 DDB(device Dependent Bitmap) 와 DIB(Device Independent Bitmap) 형으로 나뉜다. DDB는 디바이스의 설정에 따라 이미지의 표현 방식이 달라지게 되며, DIB는 디바이스에 무관하게 이미지에 내재되어 있는 설정대로 표현되는 방식이다. 예를 들어 DDB 이미지는 디바이스의 설정에 따라 흑백/컬러가 결정되지만, DIB 이미지는 이미지에 내재된 설정에 따라 흑백/컬러가 결정된다.

그림 3. Bitmap 이미지의 구조

```
typedef struct {
        unsigned short int type;                /* Magic identifier             */
        unsigned int size;                      /* File size in bytes           */
        unsigned short int reserved1, reserved2;
        unsigned int offset;                    /* Offset to image data, bytes */
} HEADER;
```

표 1. 비트맵 헤더의 자료구조

```
typedef struct {
        unsigned int size;              /* Header size in bytes      */
        int width,height;               /* Width and height of image */
        unsigned short int planes;      /* Number of colour planes   */
        unsigned short int bits;        /* Bits per pixel            */
        unsigned int compression;       /* Compression type          */
        unsigned int imagesize;         /* Image size in bytes       */
        int xresolution,yresolution;    /* Pixels per meter          */
        unsigned int ncolours;          /* Number of colours         */
        unsigned int importantcolours;  /* Important colours         */
} INFOHEADER;
```

표 2. 비트맵 인포헤더의 자료구조

예를 들어 grayscale 팔레트를 만드는 방법은 대략적으로 다음과 같다. Grayscale 은 256 가지 색만을 가지므로, 256 가지 색에 대해서만 정의를 해주면 된다. 다만, 팔레트는 기본적으로 R, G, B 요소를 모두 가지고 있으므로, 각 컬러 값에 대해 R, G, B 요소들의 값을 모두 정의해 주어야 한다. 팔레트를 구성한 뒤에는 실제 컬러 값들을 받아서 이를 grayscale로 변환해 주어야 하고, 이 변환된 값들이 팔레트에 구성된 색상표대로 화면에 디스플레이 된다.

```
…
LPRGBQUAD pDibQuad = (LPRGBQUAD) mg_lpvColorTable;
for(int i = 0; i < 256; i++) {
        pDibQuad->rgbRed = i;
        pDibQuad->rgbGreen = i;
        pDibQuad->rgbBlue = i;
        pDibQuad++;
}
```

표 3. Grayscale 팔레트 구성의 대략적인 방법

컬러 값을 grayscale로 변환 하는 방법은 여러 가지가 존재하지만, 일반적으로 NTSC(National Television System Committee)와 PAL(Phase Alternating Line)의 규격을 따라 공식 1에 의하여 변환한다. 이는 사람이 인지하는 빨강, 초록, 파랑 색에 가중치를 주어 컬러 이미지와 grayscale 이미지를 비슷하게 인지하도록 하는 목적으로 만들어진 공식이다.

$$G = 0.299R + 0.587G + 0.114B$$

식 1. NTSC 규격의 컬러 값을 grayscale로 변환하는 공식.

한편, ATSC(Advanced Television Systems Committee)에서 개발한 HDTV(High-definition television) 에서는 공식 2에 의한 변환을 사용하고, 그 외에 몇 가지 다른 변환 방식들이 존재한다.

$$G = 0.2126R + 0.7152G + 0.0722B$$

식 2. ATSC 규격의 컬러 값을 grayscale로 변환하는 공식.

16진수를 이용한 비트 연산

16 비트 이미지에서 각 컬러 값을 얻기 위하여 비트 연산을 사용하면 연산 속도도 빠르고 코드도 간결해 진다. Hexadecimal 코드는 0x 로 시작하며 16진수 0x0 ~ 0xf 는 이진수 0000 ~ 1111을 나타낸다. 비트 연산을 통하여 16 비트 값에서 R, G, B 값들을 얻어 내는 방법은 실습시간에 다루게 된다.

컬러 변환 프로그램 구현

이번 실습에서는 grayscale, 24 비트, 16 비트 비트맵 이미지들을 받아서 각각의 컬러 값들을 변환하는 프로그램을 작성해 본다. 비트맵을 구성하는 헤더, 인포헤더, 팔레트 등의 처리 루틴은 모두 주어지므로, 실질적인 픽셀 값의 처리만을 다루면 된다.

1) Grayscale 변환

a. Grayscale 이미지를 받아서, 각 픽셀 값들을 반으로 줄여서 화면에 출력한다.

b. Grayscale 이미지를 받아서, 각 픽셀 값들을 두 배로 증폭하여 화면에 출력한다. 만약 이미지가 부자연스럽게 보인다면 그 이유가 무엇인지 생각해 보고, 이를 바로 잡아 보도록 한다.

2) 24 비트 컬러 이미지 변환

24 비트 이미지를 받아서, 식 1을 이용하여 grayscale로 변환하여 화면에 출력한다.

3) 16 비트 컬러 이미지 변환

16 비트 이미지를 받아서, 식 1을 이용하여 grayscale로 변환하여 화면에 출력한다. 만약 이미지가 부자연스럽게 보인다면 그 이유가 무엇인지 생각해 보고, 이를 바로 잡아 보도록 한다.

* 24 비트 이미지의 경우, 각 24 비트마다 B, G, R 값들이 8비트 단위로 저장되며, 16 비트 이미지의 경우 각 16 비트마다 B, G, R 값들이 5 비트 단위로 저장되고 16번째 비트 값은 무시된다. 0 번째 비트가 최하위 비트이다.

숙제: 효율적인 컬러 변환 및 OpenCV를 이용한 컬러 변환

1) 위 컬러 변환에서 우리는 읽혀지는 모든 컬러 값들을 grayscale로 변환해주는 공식에 대입하였다. 이는 특정 컬러 값이 많이 나타나는 이미지에서 불필요하게 동일한 연산을 반복하는 결과를 불러온다. 이러한 문제를 해결할 수 있는 효율적인 컬러 변환 방법에 대하여 기술해보자.

2) Open source 라이브러리인 OpenCV에 컬러 변환을 수행하는 함수들이 지원된다. 이러한 함수를 사용하여 위에서 실습한 컬러 변환을 해보고, 어떤 경우에 OpenCV와 같은 툴의 사용이 제한될 수 있는지에 관하여 기술해보자. 제공되는 OpenCVcolorConversion 프로젝트를 이용한다.

OpenCV 이용하기

1. OpenCV 다운로드 및 설치
   http://sourceforge.net/projects/opencvlibrary/files/latest/download?source=files)

   OpenCV에는 여러 가지 버전이 존재하며, 필요에 따라 여러 가지 다른 버전을 사용해야 하는 경우가 있다. 따라서 각각의 OpenCV의 버전을 표시해 놓는 것이 좋다. C:₩OpenCV300에 압축을 푼다.

   시작 메뉴 → 컴퓨터 오른쪽 마우스 클릭 → 고급 시스템 설정 → 고급 탭 → 환경변수
   아래쪽 시스템 변수에서 Path 선택 → 편집: 아래의 경로를 추가한다

   vc11 = Visual Studio 2012
   vc12 = Visual Studio 2013

   C:₩OpenCV300₩opencv₩build₩x86₩vc12₩bin (Visual Studio 버전에 맞게 경로 설정)

시스템 경로 변경 후에는 비주얼 스튜디오를 재시작 하여야 변경된 경로가 인식된다.

2. OpenCV 함수를 이용하는 ColorConversion 프로젝트 생성
   New project → Win32 Condole Application → Set directory, project name → Finish

   Project Properties, C/C++ → General → Additional Include Directories 에 아래 추가
   C:₩OpenCV300₩opencv₩build₩include
   Project Properties, Linker → General → Additional Library Directories 에 아래 추가

C:₩OpenCV300₩opencv₩build₩x86₩vc12₩lib

Project Properties, Linker → Input → Additional Dependencies 에 아래 추가
opencv_ts300d.lib
opencv_world300d.lib

3. lena_8.bmp, lena_24.bmp, image_16.bmp 를 입력받아, 강의 시간에 배운 컬러 변환 작업을 OpenCV 함수들을 이용하여 실행해 본다.
cvNamedWindow, cvShowImage, imwrite, addWeighted, cvtColor 등의 함수들을 이용하게 된다.

프로젝트를 빌드 및 실행하여, 아래 네 개의 이미지들을 저장한다.
problem1_lena_8_half.bmp - lena_8.bmp 의 이미지 값을 반으로 줄인 것
problem1_lena_8_double.bmp - lena_8.bmp 의 이미지 값을 두 배로 늘린 것
problem2_lena_24_gray.bmp - lena_24.bmp 을 grayscale 로 전환한 것
problem3_image_16_gray.bmp - image_16.bmp 을 grayscale 로 전환한 것

참고 문헌

M. Petrou and C. Petrou, Image Processing: The Fundamentals, 2nd ed., Wiley
R. Gonzalez, R. E. Woods and S. L. Eddins, Digital Image Processing Using MATLAB, 2nd ed., Gatesmark Publishing