고급 소프트웨어 실습(CSE4152)

디지털 이미지

목차

- 디지털 이미지
- 비트맵 (Bitmap) 이미지의 구조
- 컬러 변환 프로그램 구현
- 실습
- 숙제: 효율적 컬러 변환, OpenCV를 이용한 컬러 변환

디지털 이미지

- 레스터(Raster) 방식
  - 일정수의 열(column)과 행(row)으로 이루어져 있음
  - 특정 열과 행 값을 픽셀(Pixel)이라 부름
  - 각 픽셀은 컬러나 밝기를 나타내는 정량화 된 값을 가짐
  - 각 픽셀 값, 또는 컬러 이미지의 경우 각 픽셀의 컬러 값은 8비트로 표현, 더 정교한 표현을 위해 16 비트 값을 사용하기도 함
  - BMP, PNG, TIFF, JPEG, GIF 등
  - 세밀한 부분까지 자유롭게 묘사가 가능한 반면, 이미지의 크기가 변함에 따라 원래의 선명함을 유지하기가 어려움
- 벡터(Vector) 방식
  - 선, 커브, 도형 등의 기본 모양을 포맷화된 수식의 형태로 저장
  - 표현할 수 있는 형상에 제약이 있지만 크기 변환에도 선명도를 유지할 수 있음
  - EPS, SVG, PDF 등

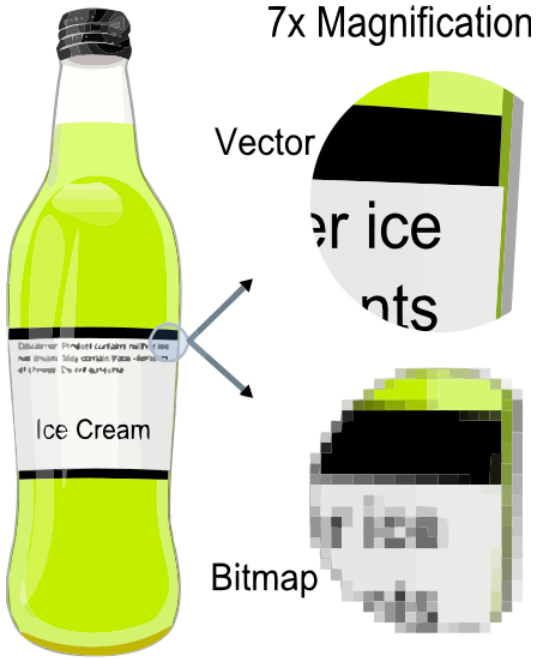

그림 1. Bitmap 이미지와 벡터 이미지의 차이.
(http://en.wikipedia.org/wiki/Vector_graphics)

디지털 이미지

- 컬러 이미지는
  - R, G, B 의 세가지 구성 요소를 가지며 이를 컬러 채널(color channel)이라고 한다

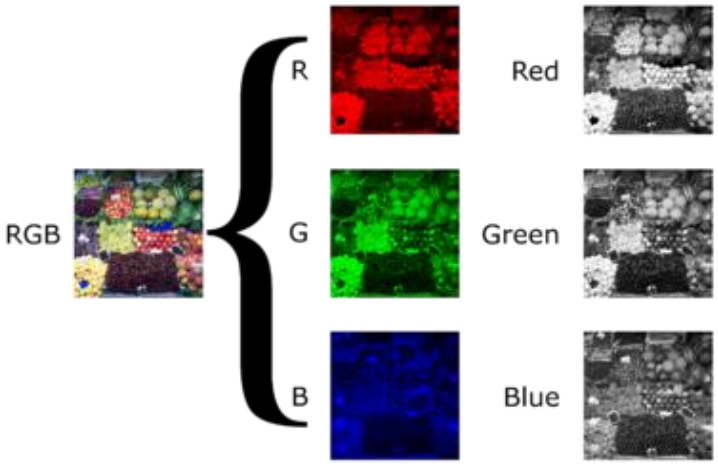

그림 2. 컬러 이미지의 R, G, B 채널
(http://en.wikipedia.org/wiki/Grayscale)

비트맵 (Bitmap) 이미지의 구조

- 비트맵 이미지 파일은 헤더, 인포헤더, 팔렛, 이미지 데이터로 구성

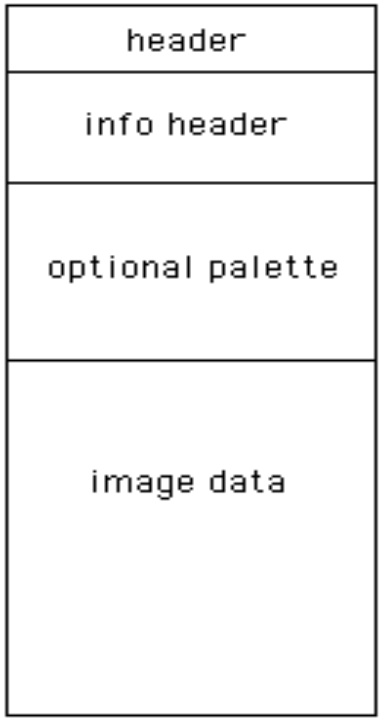

그림 3. Bitmap 이미지의 구조

```
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;
```

그림 4. 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;
```

그림 5. 비트맵 인포헤더의 자료구조

비트맵 (Bitmap) 이미지의 구조

- Gray scale (흑백, monotone) 팔레트를 만드는 방법은 아래 그림과 같다. Grayscale은 총 256 가지의 색만 존재하므로 256가지의 경우의 수만 정의해 주면 된다.
- 일반적인 컬러 이미지라면 256×256×256 가지 수의 경우를 정의해 주어야 할 것이다.

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

그림 6. Grayscale 팔레트 구성 방법

컬러 값 Grayscale로 변환

- **사람이 인지하는 빨강, 초록, 파랑 색에 가중치를 주어 컬러 이미지와 grayscale 이미지를 비슷하게 인지하도록 하는 목적.**

- NTSC(National Television System Committee)

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

식 1. NTSC 규격의 컬러 값을 grayscale로 변환하는 공식.

- ATSC(Advanced Television Systems Committee)

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

식 2. ATSC 규격의 컬러 값을 grayscale로 변환하는 공식.

16/24 비트 이미지

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

실습

- Grayscale 변환
  - Grayscale 이미지를 받아서, 각 픽셀 값들을 반으로 줄여서 출력
  - Grayscale 이미지를 받아서, 각 픽셀 값들을 2배로 늘려서 출력
- 24 비트 컬러 이미지 변환
  - 24 비트 이미지를 받아서, 식 1을 이용하여 grayscale로 변환하여 화면에 출력한다.
- 16 비트 컬러 이미지 변환
  - 16 비트 이미지를 받아서, 식 1을 이용하여 grayscale로 변환하여 화면에 출력한다.
- 결과 이미지가 부자연스럽게 나타날 경우 원인을 찾아서 수정할 것

숙제: 효율적인 컬러 변환 및 OpenCV를 이용한 컬러 변환

- 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 기술한다.

- 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 기술한다. 제공되는 OpenCVcolorConversion 프로젝트를 이용한다.

1주차 실습 및 과제 안내

- **1주차에는 OpenCVcolorConversion와 colorConversion 두 개의 프로젝트가 있습니다.**
- 실습 시간 사이버캠퍼스에 업로드
- 실습은 colorConversion 프로젝트로 진행
- 과제는 OpenCVcolorConversion 프로젝트의 완성된 코드와 함께 제출

- colorConversion 프로젝트는 Opencv를 사용하지 않아 버전과 무관하게 **정상적으로 동작하는 것을 확인**하였습니다.

- OpenCVcolorConversion **visual studio2019(opencv-4.4.0) 환경에서 정상적으로 동작하는 것을 확인하였습니다.**

- 실습 환경 설정을 위한 별도의 설명 자료를 공지사항에 업로드 했습니다.
- 실습 시간 전까지 실습할 PC에 설치하고 테스트 하길 바랍니다.
- 실습 환경 : Windows, visual studio 2017 또는 visual studio 2019(권장), Opencv 4.4.0(최신 버전)

실습/과제 제출

제출 안내

- 사이버캠퍼스 과제/실습 란을 통해 제출
- 제출 기한
  - 과제 : 다음 실습 일 전날 23:59분까지
  - Late 없음. 0점 처리함

  예시) 수요일 반의 경우,

  - 다음주 화요일 밤 11시 59분 59초까지

제출방식

- 제출 양식
  - 첨부 파일
    - ✓[고소실_O주차실습]O반_20181600_홍길동.zip
    - ✓[고소실_O주차과제]O반_20181600_홍길동.zip
    - 예시) [고소실_1주차실습]1반_20181600_홍길동.zip

** 형식 틀릴 시 감점!
*** 형식 미 준수로 인한 불이익은 본인 책임(과제 유실 우려)

첨부 파일 제출 시 유의사항

- 실습파일 제출 시 첨부파일이 대용량으로 변환된 경우, 다음과 같이 처리하여 제출한다.
  1. 빌드 → 솔루션정리 후 저장
  2. Debug 폴더 삭제
  3. ipch 폴더 삭제 – 필수!
     ex) VS2019의 경우 **.vs** 폴더 → **프로젝트 이름** 폴더 → **v16** 폴더 → **ipch** 폴더 삭제
  4. 프로젝트 폴더 자체를 압축

  - 만약 이렇게 했는데도 압축파일 크기가 30MB를 넘는다면, 실습 시 사용했던 cpp 파일만 압축하여 제출한다.

- **과제 시 코드를 제출할 경우에도 위와 동일한 방법으로 진행**