패스트캠퍼스 'OpenCV로 배우는 컴퓨터비전 프로그래밍' 커리큘럼

1주차 ● **컴퓨터 비전과 OpenCV의 만남**

- 컴퓨터 비전 프로그래밍 기본 개념 / OpenCV 개요와 설치
- OpenCV 예제 프로젝트 만들기 : HelloCV
- OpenCV 주요 함수 설명
- [실습] OpenCV 버전 출력하기/ 정지 영상 화면 출력하기/ 영상 파일 포맷 변환 프로그램

2주차 ● **OpenCV 영상 처리 기초**

- OpenCV 주요 클래스 사용법: Mat, Scalar, etc. / OpenCV 주요 기능: 관심 영역, 연산 시간 측정
- 영상의 밝기와 명암비 조절 : 기초부터 고급 이론까지
- 히스토그램 분석: 히스토그램 스트레칭과 평활화
- [실습] OpenCV 프로젝트 템플릿 사용하기 / 마스크 기능을 이용한 영상 합성 / 밝기 및 명암비 자동 보정 프로그램

3주차 ● **필터링과 기하학적 변환**

- 영상의 산술 연산
- 필터링 : 블러링, 샤프닝, 잡음 제거
- 기하학적 변환 : 어파인 변환(이동/전단/크기/회전)과 투시 변환
- [실습] 다양한 필터링 코드 구현 / 영상의 잡음 제거와 프로파일 분석 / 찌그러진 명함 영상 반듯하게 펴기

4주차 ● **에지 검출과 응용 / GPU & 병렬 프로그래밍**

- 에지 검출 : 그래디언트, 소벨, 캐니
- 허프 변환 : 직선 검출, 원 검출
- OpenCV GPU 활용 : CUDA & OpenCL(T-API) / OpenCV 병렬 프로그래밍
- [실습] 에지 검출 필터 구현 (소벨,캐니) / 자동차 주행 차선 검출 / 카메라&동영상 처리 프로그래밍 / OpenCL, 병렬 프로그래밍 속도 비교

5주차 **컬러 영상 처리 / 이진 영상 처리**

- 컬러 영상 처리: 색상 공간 이론과 활용
- 이진화 / 모폴로지 / 레이블링과 외곽선 검출
- [실습] 특정 색상 영역 추출하기 / 키보드 문자 영역 추출하기 / 다각형 인식 프로그램

6주차 **객체 검출과 추적**

- 템플릿 매칭 / 모멘트 / Cascade 분류기
- 배경 파분과 객체 추적 : 평균 이동, 캠시프트
- [실습] 인쇄체 숫자 인식 프로그램 / 실시간 얼굴 & 눈 검출 프로그램
- [실습] Project #1 : 얼굴 인식 스티커 (간단 스노우앱)

7주차 **특징점 검출과 매칭**

- 코너 검출 기법 : Harris, FAST
- 크기 불변 지역 특징점 검출 : SIFT, ORB
- 특징점 기술과 매칭 / HOG 알고리즘
- [실습] 크기 및 회전 불변 객체 검출 / 이미지 스티칭 프로그램
- [실습] Project #2 : AR 비디오 플레이어

8주차 **머신 러닝과 딥러닝**

- 머신 러닝 알고리즘: 머신 러닝 개요, K-Mean, KNN, SVM 알고리즘
- OpenCV와 딥러닝: 딥러닝과 CNN 개요, OpenCV DNN 모듈 활용법
- [실습] 필기체 숫자 인식 프로그램 / 유명한 딥러닝 모델 실행 예제 : GoogLeNet, YOLO, OpenPose / 딥러닝 기반 얼굴 검출 및 인식 프로그램 (Train & Test)

OpenCV란?

- Open Source Computer Vision Library. 오픈소스로 개발되고 있는 컴퓨터 비전 및 머신러닝 라이브러리
- 영상파일 입출력, 영상 화질 향상, 객체 검출과 인식, 추적, 3차원 비전 문제 해결 등 제공
- 기본적으로 C/C++언어로 작성됨. Python, Java, Matlab 등 인터페이스도 제공

- 다수의 모듈로 구성.
- 모듈은 OpenCV에서 제공하는 다양한 클래스와 함수를 기능과 성격에 따라 모아 놓은 OpenCV의 부분 라이브러리
- 핵심 클래스와 함수는 core모듈, 영상 출력 기능은 highgui모듈에 들어있음.
- 여러 모듈 중 자신에게 필요한 모듈을 모두 선택하여 프로젝트에 포함시키기.
- 예시 : #include <opencv2/highgui.hpp>

설치 : https://velog.io/@lcooldong/C-OpenCV-%EC%84%A4%EC%B9%98

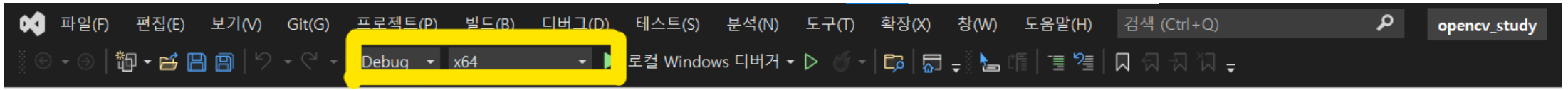
모듈 이름	설명
calib3d	카메라 캘리브레이션과 3차원 재구성
core	행렬, 벡터 등 OpenCV 핵심 클래스와 연산 함수
dnn	심층 신경망 기능
features2d	2차원 특징 추출과 특징 벡터 기술, 매칭 방법
flann	다차원 공간에서 빠른 최근방 이웃 검색
highgui	영상의 화면 출력, 마우스 이벤트 처리 등 사용자 인터페이스
imgcodecs	영상 파일 입출력
imgproc	필터링, 기하학적 변환, 색 공간 변환 등 영상 처리 기능
ml	통계적 분류, 회기 등 머신 러닝 알고리즘
objdetect	얼굴, 보행자 검출 등 객체 검출
photo	HDR, 잡음 제거 등 사진 처리 기능
stitching	영상 이어 붙이기
video	옵티컬 플로우, 배경 차분 등 동영상 처리 기술
videoio	동영상 파일 입출력
world	여러 OpenCV 모듈을 포함하는 하나의 통합 모듈

OpenCV 프로젝트 만들기

참고 : https://velog.io/@mouse0429/openCVVisual-Studio-OpenCV-%EC%84%A4%EC%B9%98

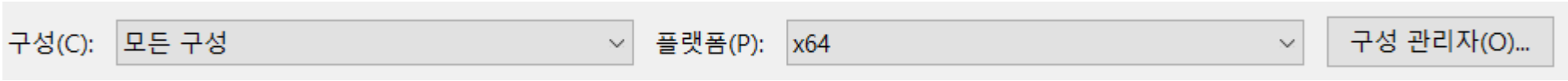
① Visual Studio 실행 → 빈 프로젝트 → 프로젝트 이름 및 경로 설정 → 소스파일 추가(.cpp)

② 툴바 솔루션 플랫폼 x64로 변경



③ 상단 바 프로젝트 → 속성

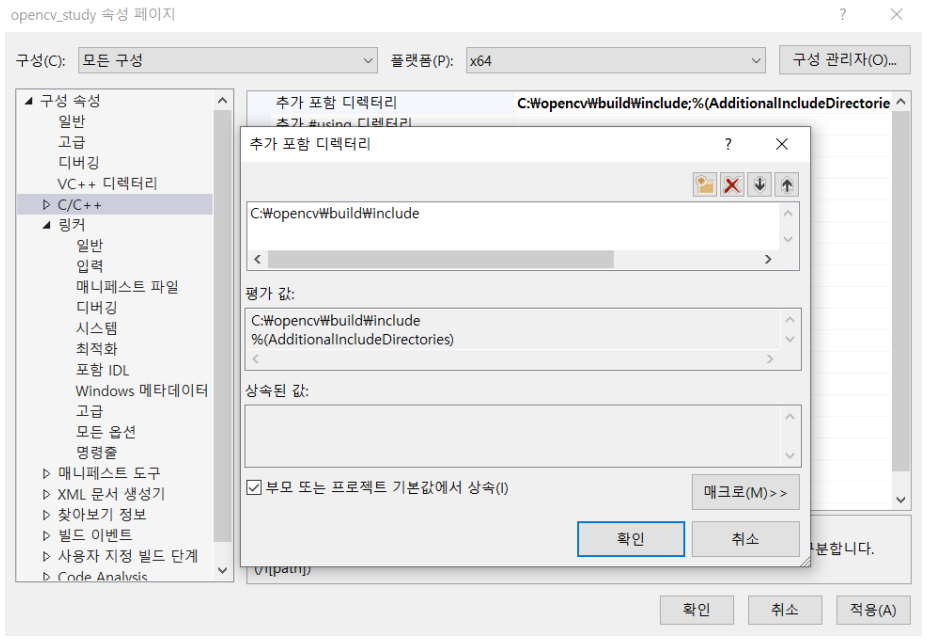
④ 구성 → '모든 구성'으로 변경



OpenCV 프로젝트 만들기

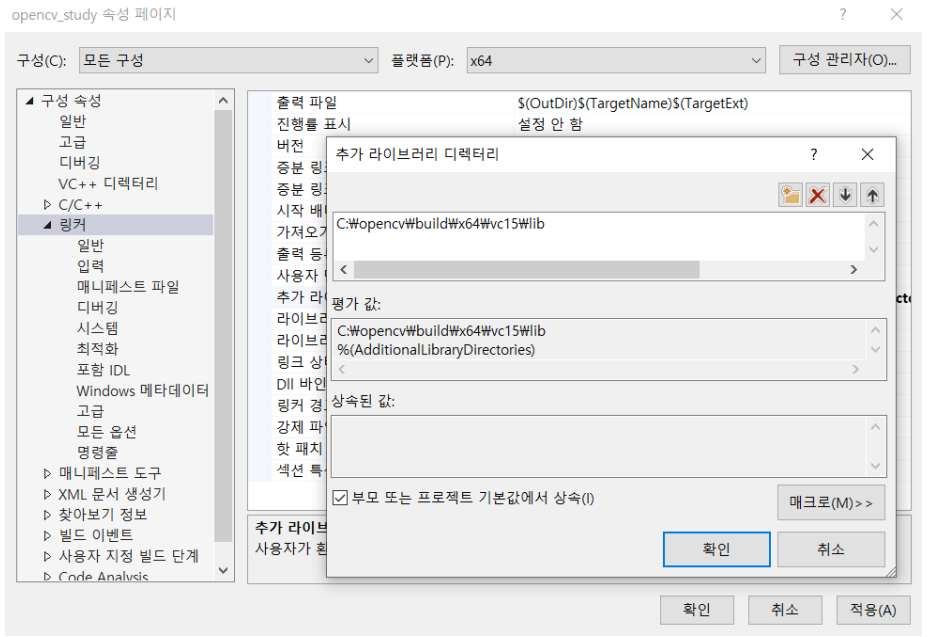
⑤ 추가 포함 디렉터리에 OpenCV include 폴더 삽입

- 구성 속성 → C/C++ → 일반 → 추가 포함 디렉터리 → 편집 → 저장경로 :₩opencv₩build₩include → '부모 또는 프로젝트 기본값에서 상속' 체크



⑥ 추가 라이브러리 디렉터리에 OpenCV 라이브러리 파일 디렉터리 지정

- 구성속성 →링커 →추가 라이브러리 디렉터리 → 편집 → 저장경로 : ₩opencv₩build₩x64₩vc15₩lib → '부모 또는 프로젝트 기본값에서 상속' 체크



OpenCV 프로젝트 만들기

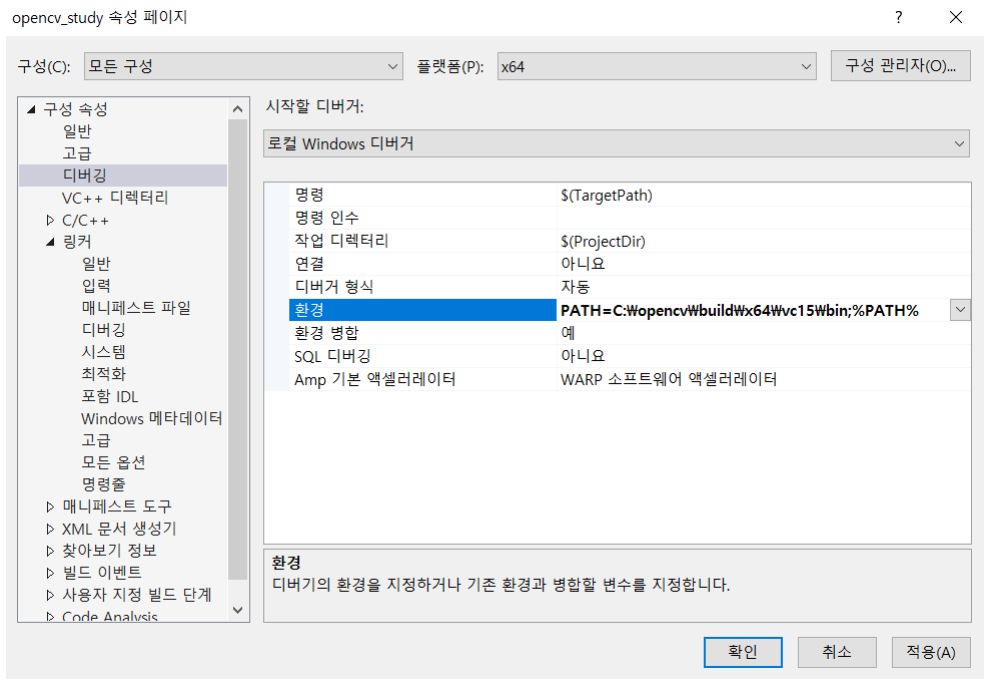
⑦ 디버깅 환경에 OpenCV 라이브러리 DLL파일 위치 추가

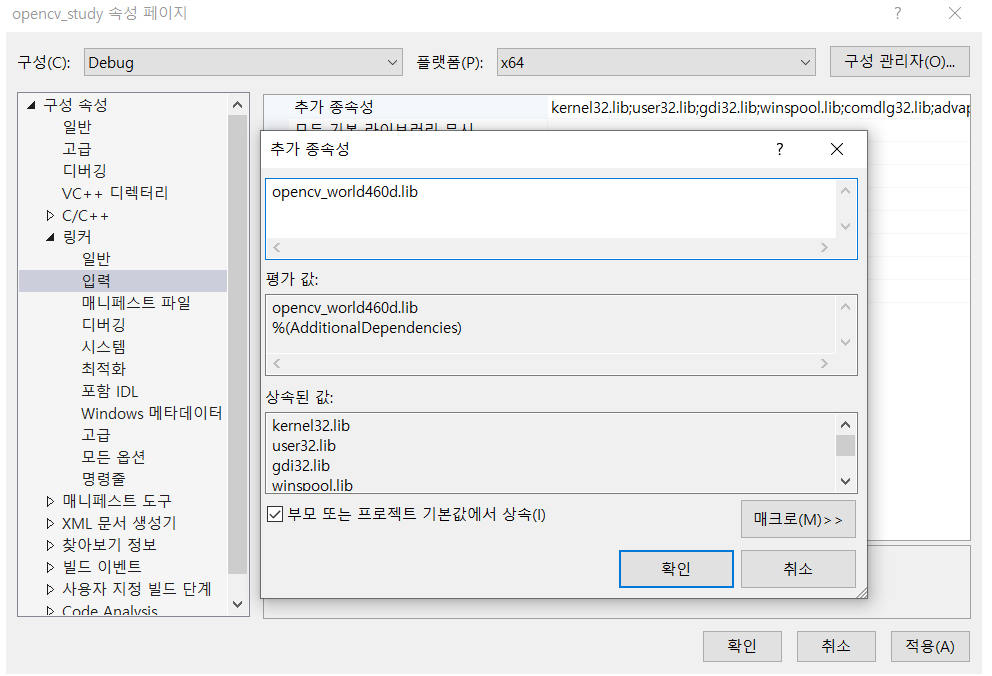
- 구성속성 → 디버깅 → 환경 → 저장경로 : ₩opencv₩build₩x64₩vc15₩bin;%PATH%

⑧ 모든 구성 → Debug로 변경 → 저장

⑨ 구성속성 → 링커 → 입력 → 추가 종속성 → 편집 → opencv_world460d.lib 적기. (버전에 따라 4.4.0이면 440, 4.6.0이면 460 적기) → '부모 또는 프로젝트 기본값에서 상속' 체크

⑩ 구성 Debug → Release로 변경 → 저장

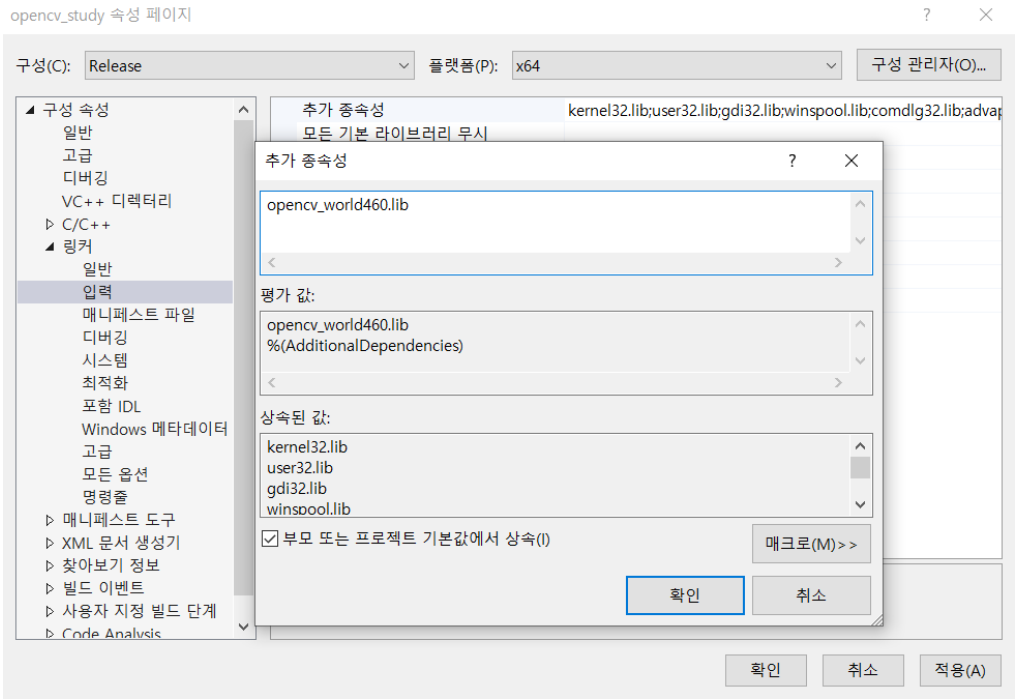




OpenCV 프로젝트 만들기

⑪ 추가 종속성 설정

- 구성 속성 → 링커 → 입력 → 추가 종속성 → 편집 → opencv_world460.lib 적기. (버전에 따라 4.4.0이면 440, 4.6.0이면 460 적기, d적으면 안됨!) → '부모 또는 프로젝트 기본값에 서 상속' 체크

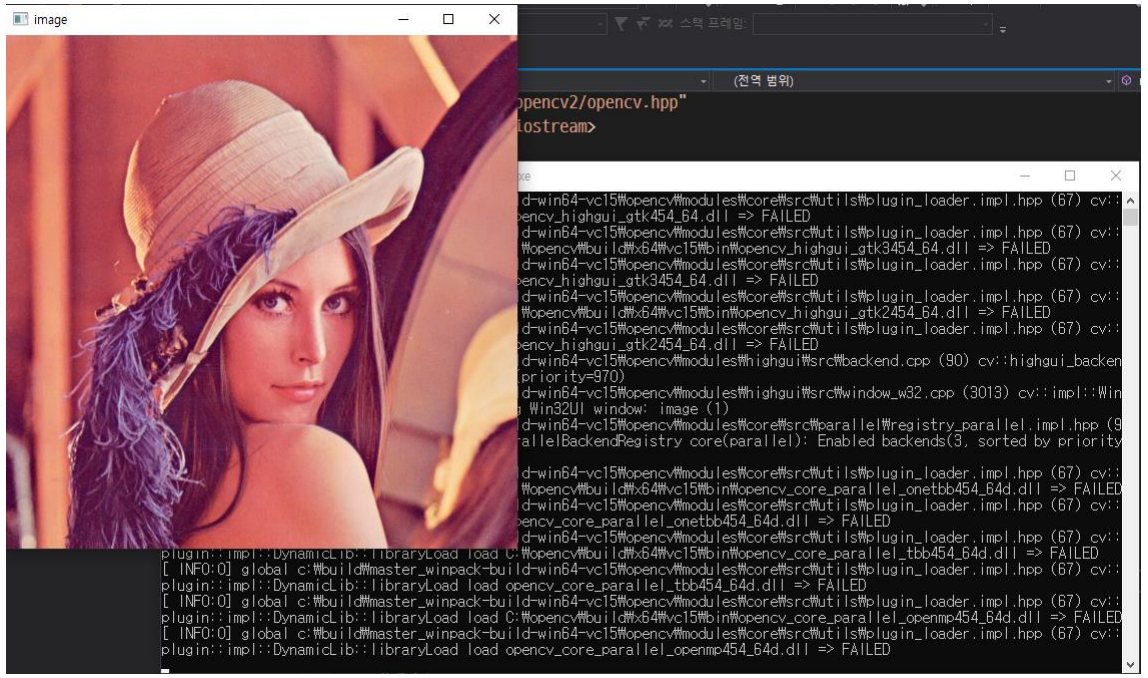


환경 설정 끝!

OpenCV 프로젝트 만들기 : OpenCV버전 출력 & 영상 출력

```cpp
#include "opencv2/opencv.hpp"
// OpenCV 에서 사용하는 다양한 클래스와 함수가 선언되어 있는 헤더파일
#include <iostream>

using namespace cv;
using namespace std;

int main()
{

	cout << "Hello OpenCV " << CV_VERSION << endl;
    // OpenCV 라이브러리의 버전 출력

	Mat img;
    // Mat 클래스 객체 생성. OpenCV 에선 영상 데이터를 범용 행렬 클래

	img = imread("lenna.bmp");
    // imread 함수를 사용하여 파일을 불러와 img 객체에 저장

	if (img.empty()) {
		cerr << "Image load failed!" << endl;
		return -1;
	}
    // 파일 불러오기 실패했을 때의 예외 처리 코드

	namedWindow("image");
    // namedWindow 함수를 이용하여 새로운 창 생성. 창의 이름은 image

	imshow("image", img);
    // imshow 함수를 사용하여 image라는 창에 img 객체가 갖고있는 영상

	waitKey(0);
    // 사용자의 키보드 입력을 기다리는 함수.
    // 매개변수의 디폴트값은 0으로 무한대기. 특정 키 입력을 기다리는 것

	return 0;
}
```

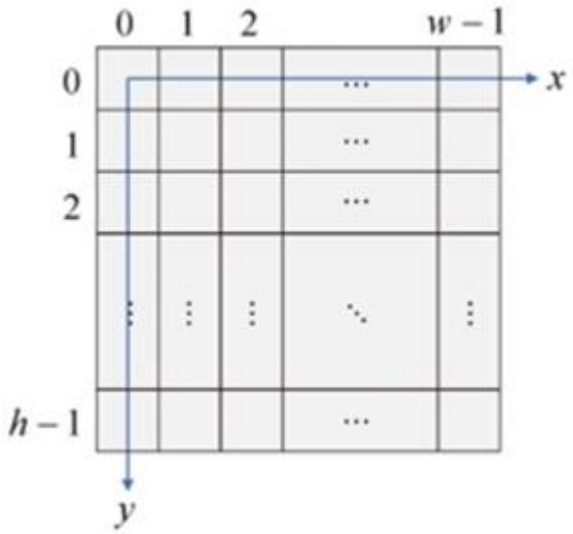
*프로젝트 폴더 내에 lenna.bmp 파일 다운 받기



```cpp
#include "opencv2/opencv.hpp"
// OpenCV 에서 사용하는 다양한 클래스와 함수가 선언되어 있는 헤더파일
#include <iostream>

using namespace cv;
using namespace std;

int main()
{

        cout << "Hello OpenCV " << CV_VERSION << endl;
    // OpenCV 라이브러리의 버전 출력

        Mat img;
    // Mat 클래스 객체 생성. OpenCV 에선 영상 데이터를 범용 행렬 클래스

        img = imread("lenna.bmp");
    // imread 함수를 사용하여 파일을 불러와 img 객체에 저장

        if (img.empty()) {
                cerr << "Image load failed!" << endl;
                return -1;
        }
    // 파일 불러오기 실패했을 때의 예외 처리 코드

        namedWindow("image");
    // namedWindow 함수를 이용하여 새로운 창 생성. 창의 이름은 image

        imshow("image", img);
    // imshow 함수를 사용하여 image라는 창에 img 객체가 갖고있는 영상

        waitKey(0);
    // 사용자의 키보드 입력을 기다리는 함수.
    // 매개변수의 디폴트값은 0으로 무한대기. 특정 키 입력을 기다리는 것

        return 0;
}
```

- imread() 함수 : image read. 영상을 읽어오는 함수
- nameWindow()함수 : 영상을 화면에 나타내기 위한 새로운 창을 생성하는 함수
- imshow()함수 : 영상을 보여주는 함수
- waitKey() 함수 : 사용자의 키보드 입력을 기다리는 함수. 사용자가 키보드를 누르기 전가지 영상을 화면에 나타나게 해준다.

- Mat 클래스 : OpenCV에서 영상 데이터는 Mat클래스를 이용하여 표현한다. 다양한 자료형의 행렬을 표현할 수 있는 범용 행렬 클래스이다. 영상은 원소가 0부터 255사이의 정수 값을 가질 수 있는 이차원 행렬이기 때문에 영상도 Mat클래스를 이용해 표현할 수 있다.
- 즉, BMP파일에 저장된 영상을 출력하기 위해서는 일단 Mat객체를 생성하고, Mat객체를 화면에 출력하는 OpenCV함수를 호출하면 된다.



- 영상을 구성하는 최소 단위 : 픽셀
- 픽셀이 모여서 2차원 영상을 구성한다.

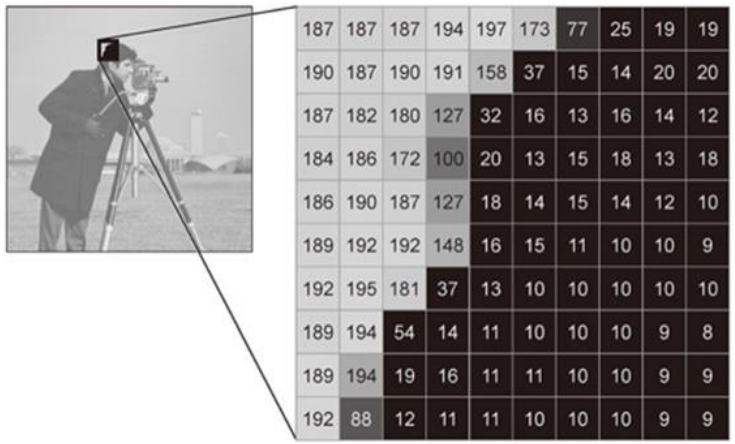
→ 영상은 픽셀이 바둑판처럼 균일한 격자 형태로 배열되어 있는 형태를 표현한다.

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix}$$

→ 2차원 평면 위에 픽셀 값이 나열된 형태이기 때문에 영상을 2차원 행렬로 표현할 수 있다.

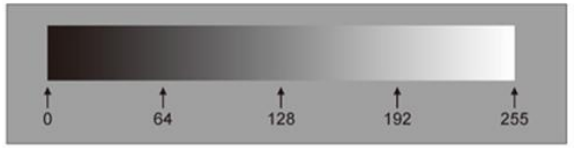
- 그레이스케일 영상 : 흑백 사진처럼 오직 밝기 정보만으로 구성된 영상. 회색조 영상
- 트루컬러 영상 : 컬러 사진처럼 다양한 색상을 표현할 수 있는 영상.

↓그레이스케일 영상에서 하나의 픽셀은 0부터 255 사이의 정수값을 가질 수 있으며, 0은 가장 어두운 검은색, 255는 가장 밝은 흰색을 표현한다.

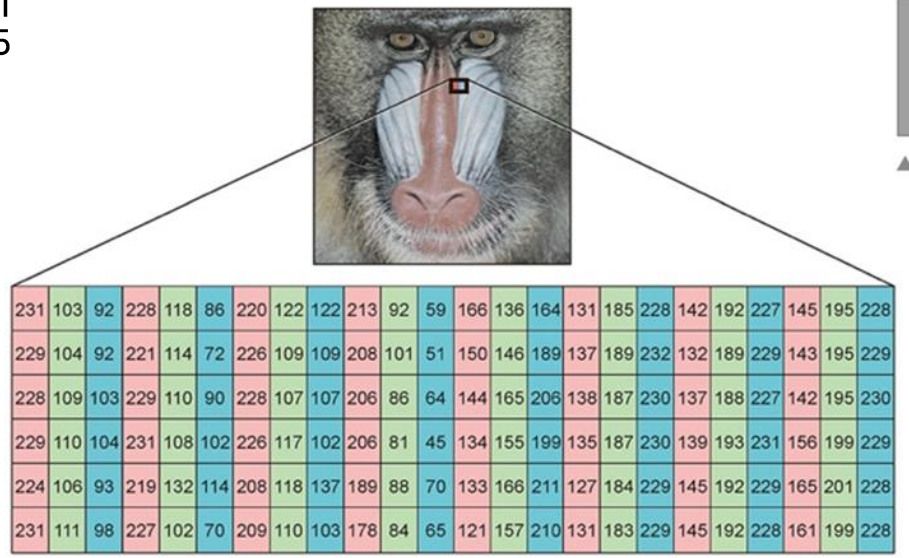


187	187	187	194	197	173	77	25	19	19
190	187	190	191	158	37	15	14	20	20
187	182	180	127	32	16	13	16	14	12
184	186	172	100	20	13	15	18	13	18
186	190	187	127	18	14	15	14	12	10
189	192	192	148	16	15	11	10	10	9
192	195	181	37	13	10	10	10	10	10
189	194	54	14	11	10	10	10	9	8
189	194	19	16	11	11	10	10	9	9
192	88	12	11	11	10	10	10	9	9

▲ 그림 1-7 그레이스케일 영상에서 픽셀 값 분포



▲ 그림 1-6 그레이스케일 값에 따른 밝기 변화



231	103	92	228	118	86	220	122	122	213	92	59	166	136	164	131	185	228	142	192	227	145	195	228
229	104	92	221	114	72	226	109	109	208	101	51	150	146	189	137	189	232	132	189	229	143	195	229
228	109	103	229	110	90	228	107	107	206	86	64	144	165	206	138	187	230	137	188	227	142	195	230
229	110	104	231	108	102	226	117	102	206	81	45	134	155	199	135	187	230	139	193	231	156	199	229
224	106	93	219	132	114	208	118	137	189	88	70	133	166	211	127	184	229	145	192	229	165	201	228
231	111	98	227	102	70	209	110	103	178	84	65	121	157	210	131	183	229	145	192	228	161	199	228

▲ 그림 1-9 트루컬러 영상에서 픽셀 값 분포

→ 트루컬러 영상의 각 픽셀은 각각 0에서 255사이의 값을 갖는 R, G, B 세 색상 성분의 조합으로 표현된다.

미니 프로젝트