ICE4027 Digital Image Processing

Lab.2 – Point Processing and Histogram

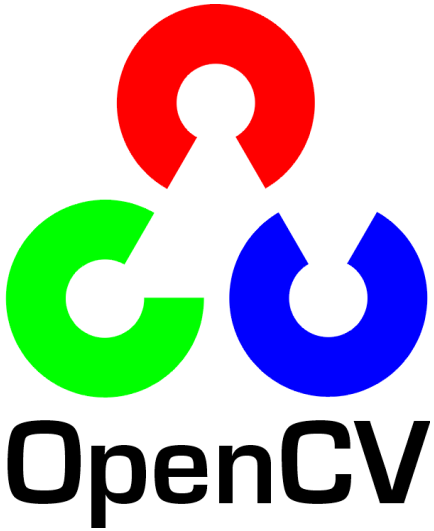
Prof. In Kyu Park

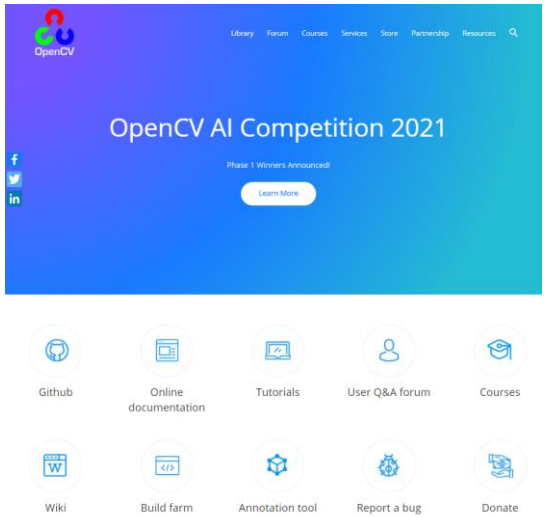
인하대학교 시각컴퓨팅 및 학습 연구실
Visual Computing and Learning Laboratory

Contents

Introduction

- **OpenCV**
  - Intel에서 개발을 주도한 오픈소스 컴퓨터 비전 라이브러리
  - 영상처리, 컴퓨터 비전에 에 필요한 기본적인 기능들을 거의 대부분 포함
  - 다양한 플랫폼 지원(Windows, Linux, Android 등)
  - 다양한 언어 지원(C/C++, Python, Java 등)

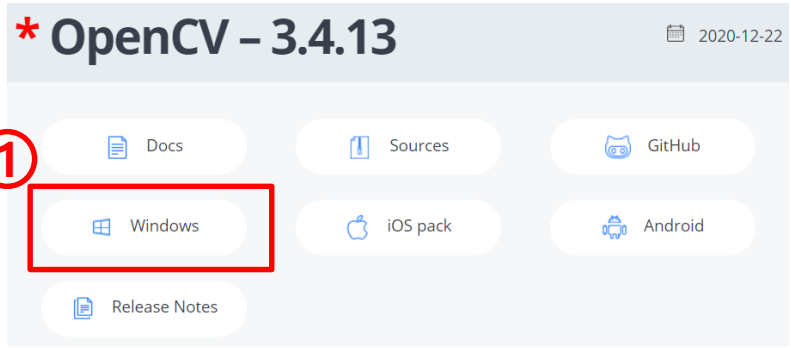


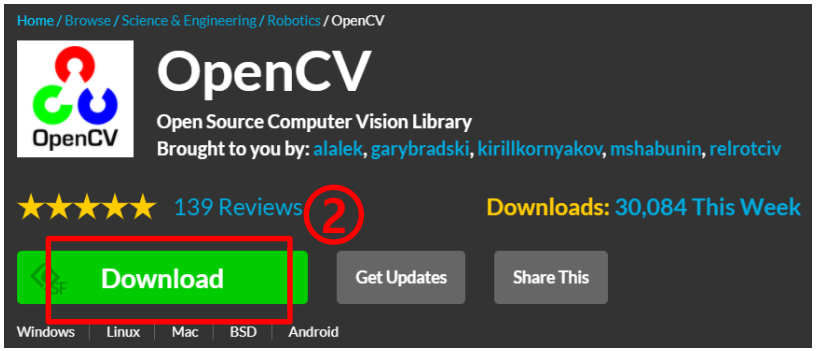


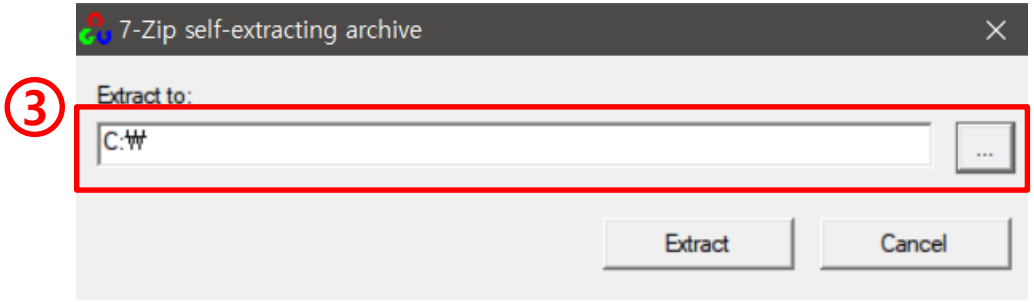
공식 홈페이지: https://opencv.org/

Installation

- 기본 모듈만 설치 시
  - OpenCV 홈페이지에서 설치파일(.exe) 다운로드 후 설치
  - 버전은 3.x, 4.x를 권장 (본 실습에서는 3을 기준으로 설명)
  - Extract 위치는 C드라이브를 권장



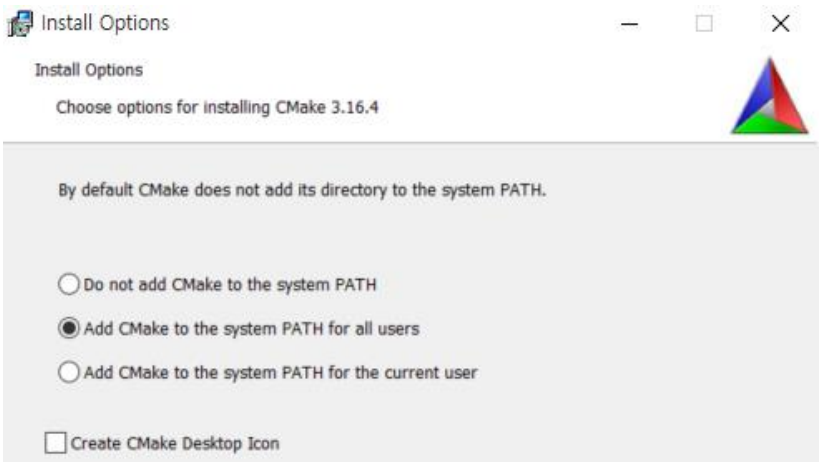




Installation

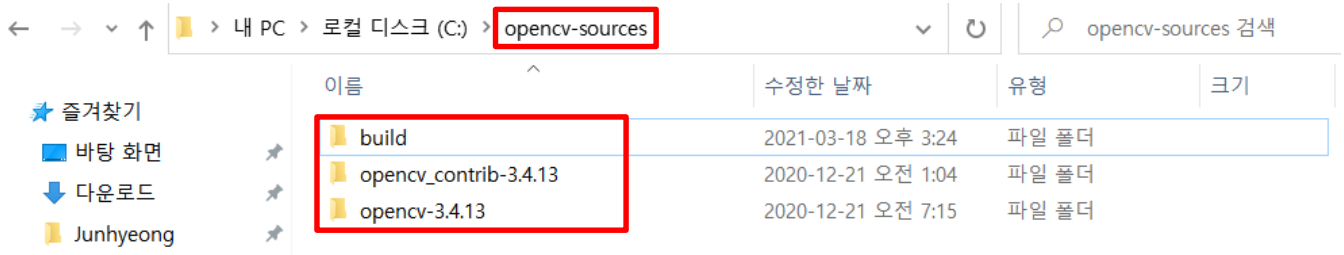
- **Extra 모듈 + nonfree 모듈 설치 시 (1)**
  - 저작권 문제 등으로 기본 모듈에 포함되기 어려운 모듈은 소스코드를 받아 CMake를 통해 빌드하면 사용할 수 있음
  - 빌드를 수행해 줄 CMake를 최신 버전으로 설치
    - https://cmake.org/download/
    - 64비트 버전인지 반드시 확인

Platform	Files
Windows x64 Installer: **Installer tool has changed. Uninstall CMake 3.4 or lower first!**	cmake-3.20.0-rc5-windows-x86_64.msi

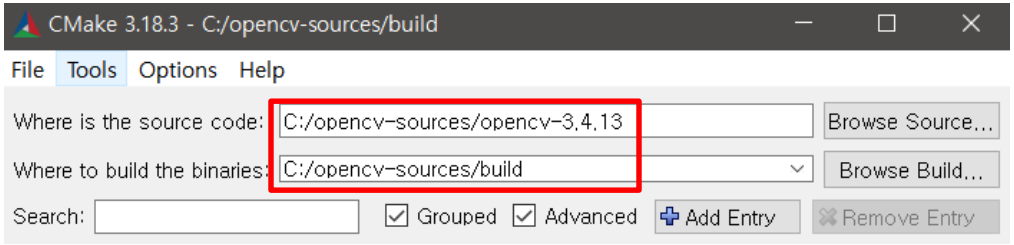


Installation

- **Extra 모듈 + nonfree 모듈 설치 시 (2)**
  - 빌드를 수행할 기본 모듈, contib모듈 소스코드를 OpenCV 공식 GitHub에서 다운로드 (contrib모듈 = extra + nonfree)
    - 기본 모듈: https://github.com/opencv/opencv/archive/3.4.13.zip
    - Contib모듈: https://github.com/opencv/opencv_contrib/archive/3.4.13.zip
  - 두 모듈 소스코드 압축을 풀고 다음과 같이 폴더 생성 및 이동

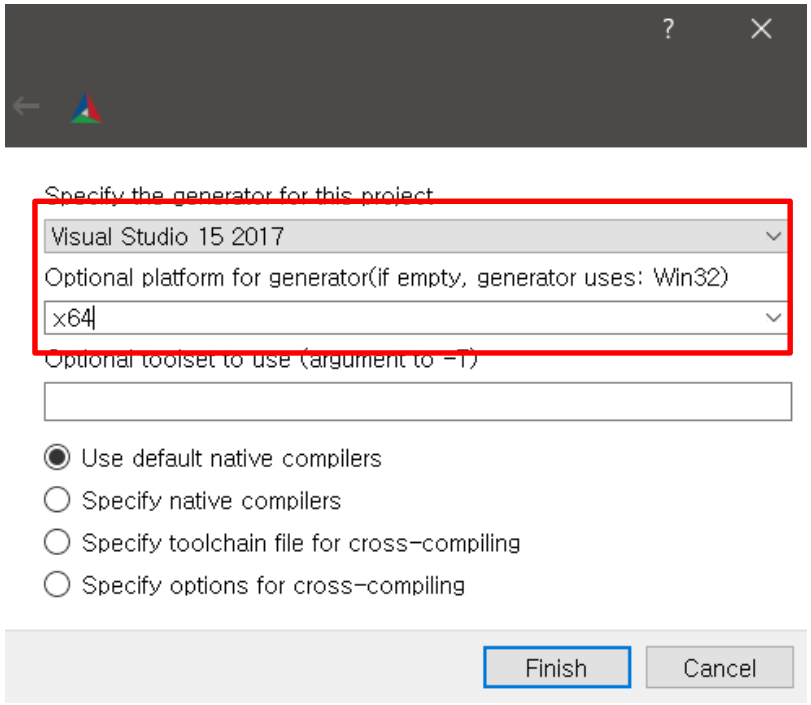


  - CMake를 관리자 권한으로 실행 후 소스코드 위치, 임시 빌드 위치 설정



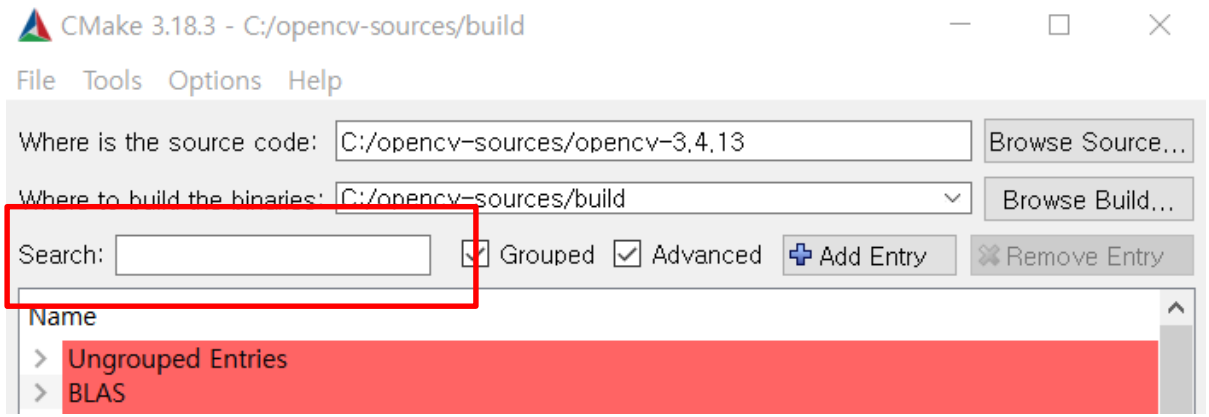
Installation

- **Extra 모듈 + nonfree 모듈 설치 시 (3)**
  - 설정을 적용하기 위해 Configure 클릭
  - 팝업 되는 창에서 다음과 같이 설정을 맞춤
    - Visual Studio는 각자 버전에 맞게 선택 (2017이상 추천)
    - 플랫폼은 반드시 64비트로 설정(x64)



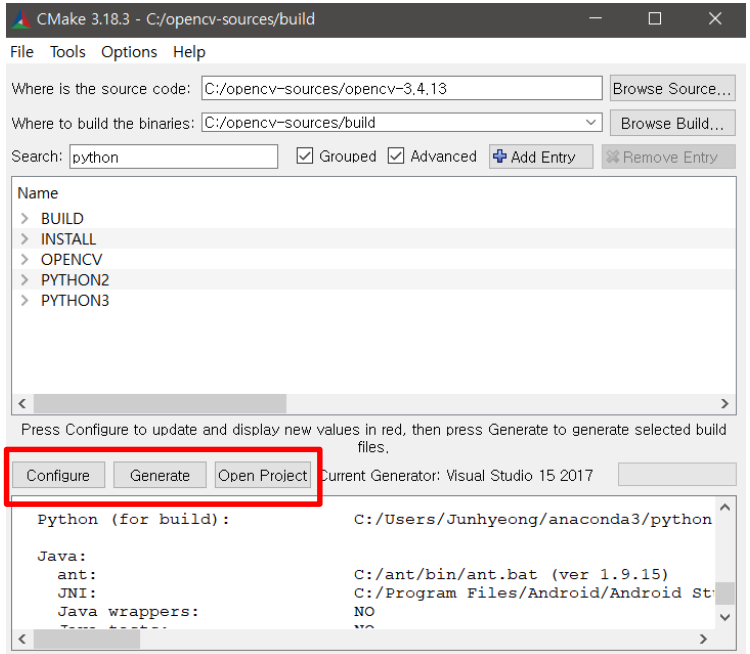
Installation

- **Extra 모듈 + nonfree 모듈 설치 시 (4)**
  - 1차적으로 configure가 완료되면 다음과 같은 내용을 search해 옵션 변경
    - BUILD_PERF_TESTS, BUILD_TESTS의 value 체크 해제
    - OPENCV_EXTRA_MODULES_PATH의 value를 C:/opencv-sources/opencv_contrib-3.4.13/modules 로 지정(contrib 모듈 위치)
    - Nonfree 모듈 사용을 위해 OPENCV_ENABLE_NONFREE의 value를 체크
    - CMAKE_INSTALL_PREFIX의 value를 C:/opencv-3.4.13/build (최종적으로 빌드된 결과가 생성되는 곳 임)
    - BUILD_PACKAGE value 체크 해제
    - WITH_1394, WITH_GSTREAMER, WITH_LAPACK, WITH_VTK의 value 체크 해제



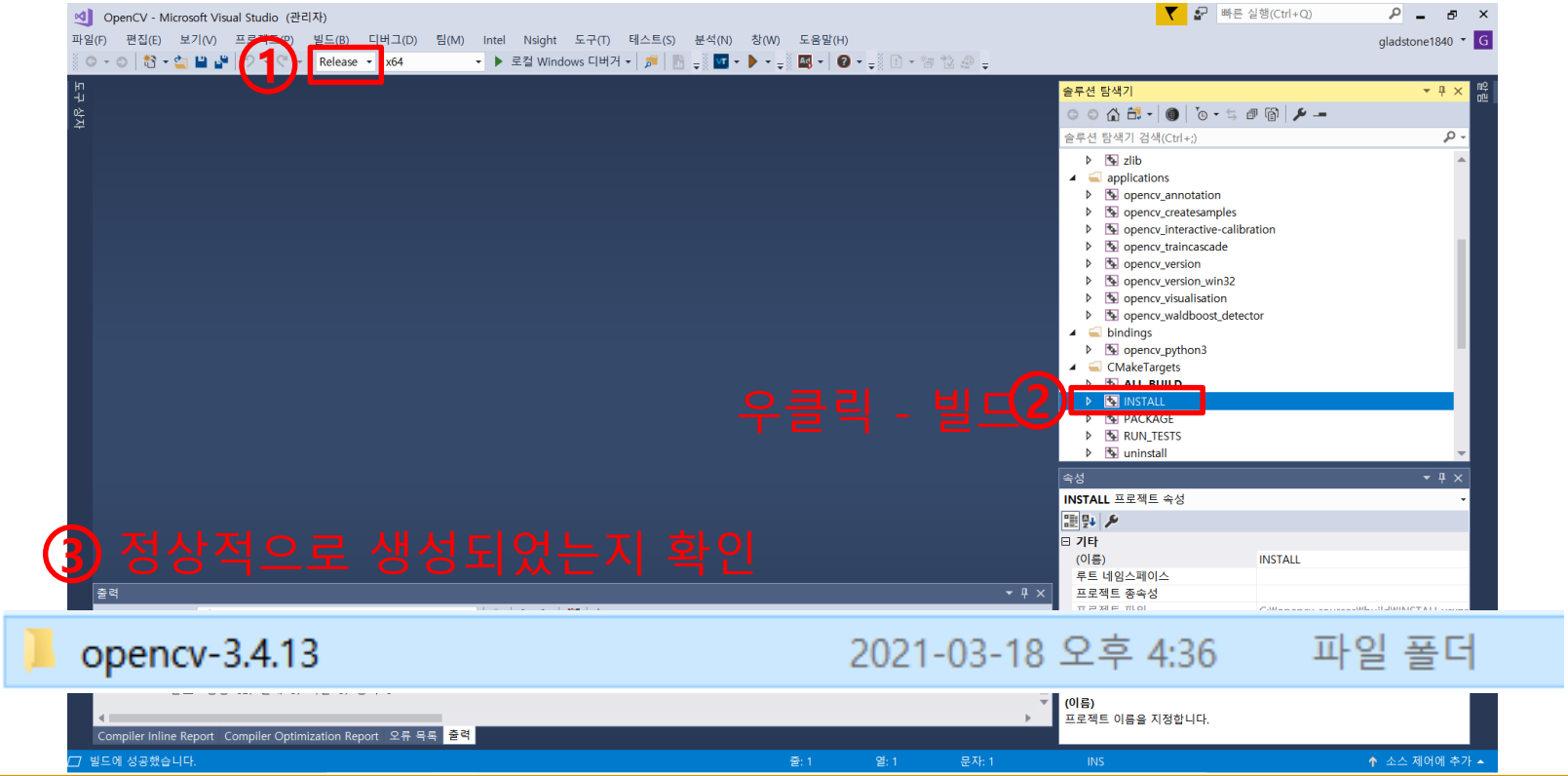
Installation

- **Extra 모듈 + nonfree 모듈 설치 시 (5)**
  - Configure를 다시 수행 후 새로 나타나는 옵션을 다음과 같이 변경
    - 모듈을 하나로 통합해 생성하기 위해 BUILD_opencv_world의 value를 체크
  - 최종적으로 다시 configure
  - Generate 클릭
  - Open Project 클릭



Installation

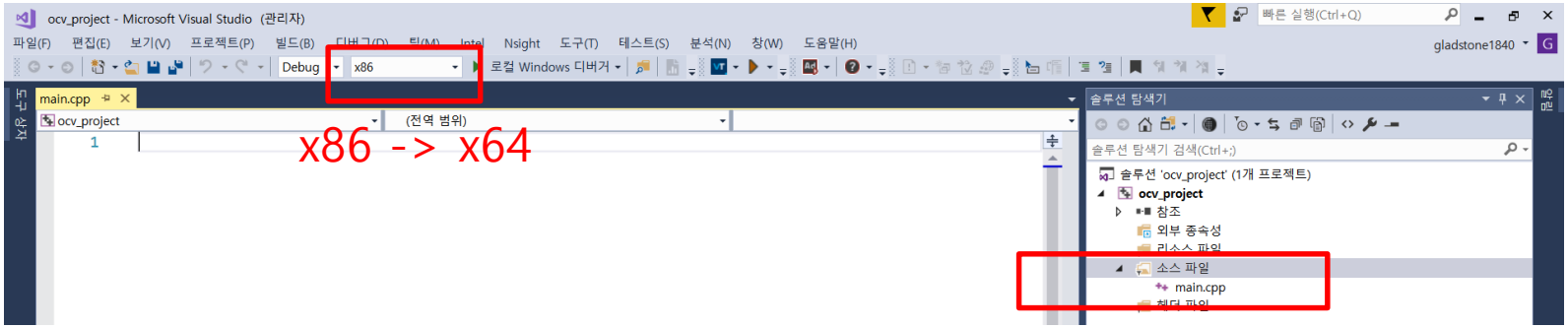
- **Extra 모듈 + nonfree 모듈 설치 시 (6)**
  - Visual Studio가 열리면 솔루션 구성을 Release로 변경한 후 INSTALL 프로젝트를 우 클릭해 빌드를 선택
  - 빌드가 완료되면 솔루션 구성을 Debug로 변경한 후 동일한 과정 반복



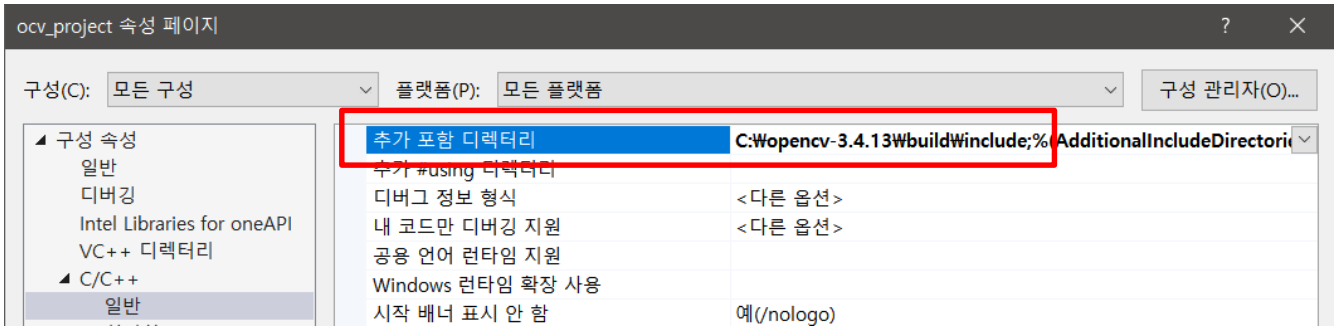
Installation

Visual Studio에서 OpenCV 사용 설정하기 (1)

- 빈 프로젝트를 생성하고 플랫폼을 x64로 변경 후 .cpp 소스 파일을 하나 생성함

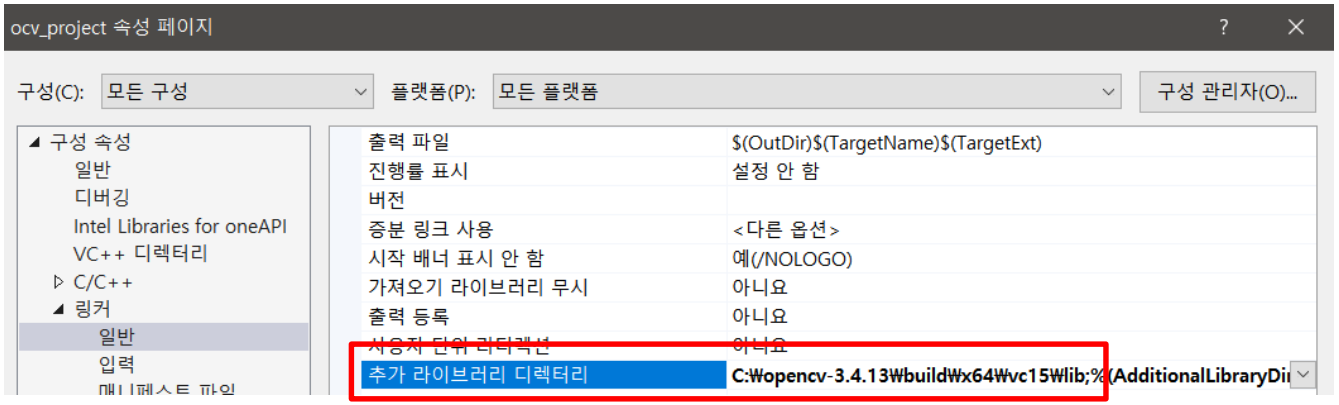


- '프로젝트-[프로젝트 이름] 속성' 클릭
- '구성-모든 구성'으로 변경
- 추가 포함 디렉터리 설정

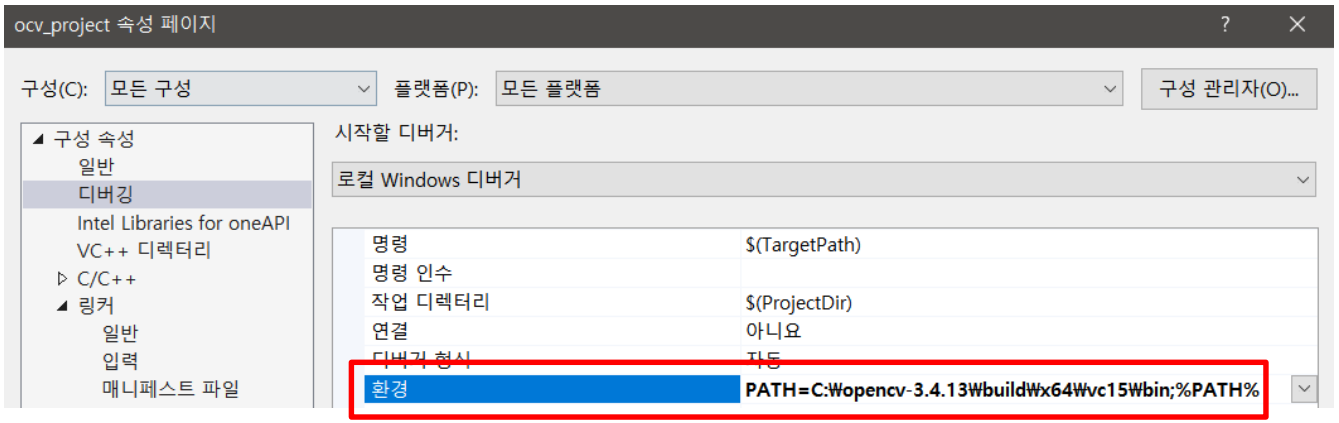


Installation

- **Visual Studio에서 OpenCV 사용 설정하기 (2)**
  - 추가 라이브러리 디렉터리 설정

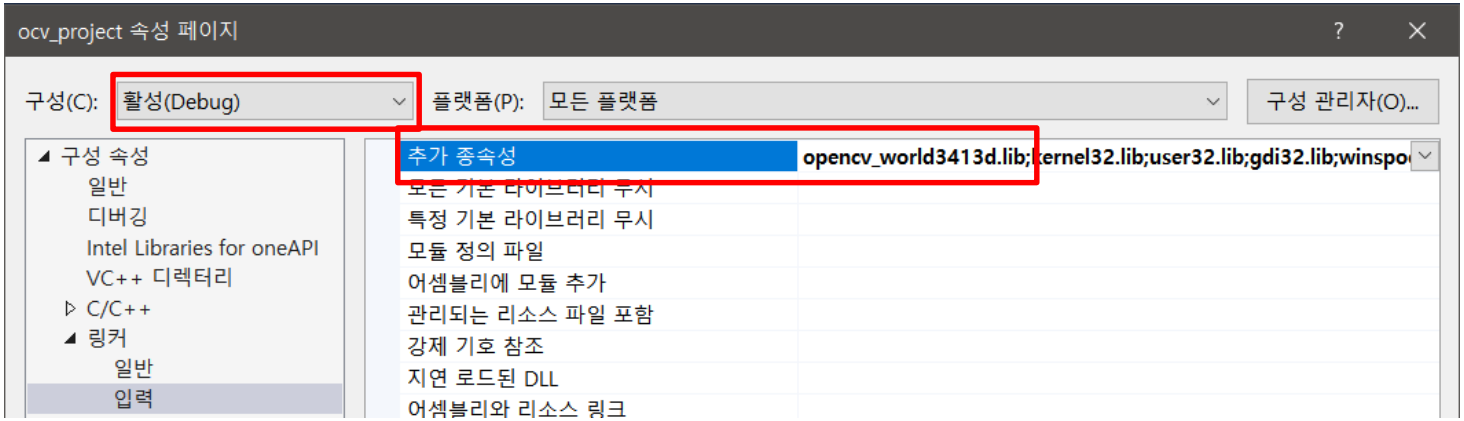


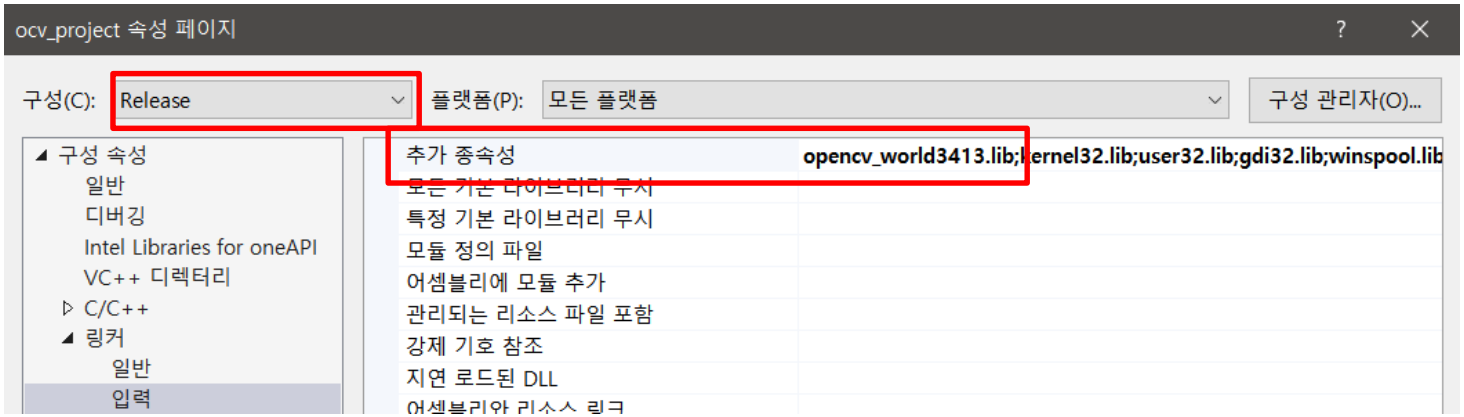
  - 환경(DLL) 설정



Installation

- **Visual Studio에서 OpenCV 사용 설정하기 (3)**
  - 추가 종속성 설정(Debug, Release 각각)





API Searching

- **OpenCV API docs**
  - OpenCV에서 제공하는 클래스, 구조체, 함수 등이 굉장히 방대하기 때문에 이들에 대해 문서화를 한 자료가 버전별로 정리되어 있음
  - 이해를 돕기 위한 각종 예제 자료들도 포함되어 있음
  - 본 실습에서 소개되는 클래스나 함수의 모든 변수, 플래그 등을 일일이 설명할 수 없기 때문에 API docs를 기준으로 찾아서 수행할 수 있도록 설명
  - API docs 링크: https://docs.opencv.org/3.4.13/index.html

Mat and I/O

- **Mat 클래스**
  - OpenCV에서 이미지와 이미지의 정보(사이즈 등)를 담게 되는 클래스
- **입출력 함수**
  - 이미지를 불러오거나 저장하는 함수
  - 기본적으로 Mat 객체로 반환하거나 Mat 객체를 입력으로 받아 수행

```
Mat cv::imread ( const String &  filename,
                 int             flags = IMREAD_COLOR
               )

void cv::imshow ( const String &  winname,
                  InputArray      mat
                )

bool cv::imwrite ( const String &              filename,
                   InputArray                  img,
                   const std::vector< int > &  params = std::vector< int >()
                 )
```

I/O and Mat

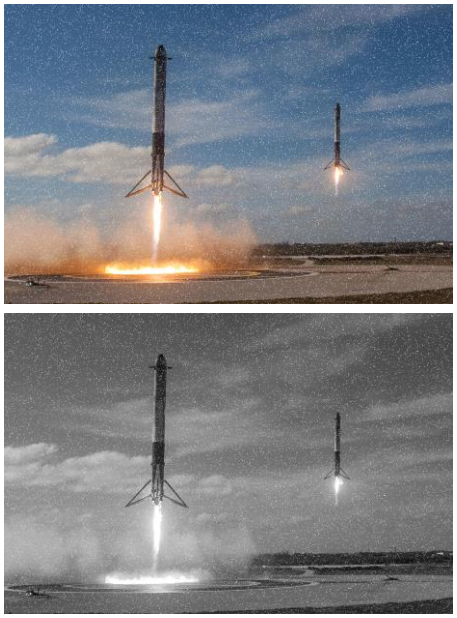
```
#include <iostream>
#include "opencv2/core/core.hpp" // Mat class와 각종 data structure 및 산술 루틴을 포함하는 헤더
#include "opencv2/highgui/highgui.hpp" // GUI와 관련된 요소를 포함하는 헤더(imshow 등)
#include "opencv2/imgproc/imgproc.hpp" // 각종 이미지 처리 함수를 포함하는 헤더

using namespace cv;
using namespace std;

int main() {                                    경로와 파일 확장자에 주의!
        Mat src_img = imread("landing.jpg", 0); // 이미지 읽기
        // int flags = IMREAD_COLOR 또는 int flags = 1 -> 컬러 영상으로 읽음
        // int flags = IMREAD_GRAYSCALE 또는 int flags = 0 -> 흑백 영상으로 읽음
        // int flags = IMREAD_UNCHANGED  또는 int flags = -1 -> 원본 영상의 형식대로 읽음

        imshow("Test window", src_img); // 이미지 출력
        waitKey(0); // 키 입력 대기(0: 키가 입력될 때 까지 프로그램 멈춤)
        destroyWindow("Test window"); // 이미지 출력창 종료

        imwrite("langding_gray.png", src_img); // 이미지 쓰기

        return 0;
}
```

Pixel Value

픽셀 값 접근 방법

- 이미지의 각 픽셀을 개별적으로 접근하는 경우
- OpenCV의 함수들을 사용하게 되면 사용할 경우가 없으나 픽셀 값을 각각 읽어 원하는 기능을 구현하거나 분석할 때 반드시 필요함
- OpenCV의 컬러 mat는 BGR 채널 순서를 가지고 있음에 유의

```
void SpreadSalts(Mat img, int num) {
        // num: 점을 찍을 개수
        for (int n = 0; n < num; n++) {
                int x = rand() % img.cols; // img.cols는 이미지의 폭 정보를 저장
                int y = rand() % img.rows; // img.rows는 이미지의 높이 정보를 저장
                /*
                나머지는 나누는 수를 넘을 수 없으므로 무작위 위치가
                이미지의 크기를 벗어나지 않도록 제한하는 역할을 하여줌
                */

                if (img.channels() == 1) {
                        // img.channels()는 이미지의 채널 수를 반환
                        img.at<uchar>(y, x) = 255; // 단일 채널 접근
                }
                else {
                        img.at<Vec3b>(y, x)[0] = 255; // Blue 채널 접근
                        img.at<Vec3b>(y, x)[1] = 255; // Green 채널 접근
                        img.at<Vec3b>(y, x)[2] = 255; // Red 채널 접근
                }
        }
}
```



수행 결과

Histogram

히스토그램 분석

- 픽셀 값의 분포를 나타낸 그래프라고 할 수 있음
- 가능한 모든 픽셀 레벨에 대한 카운트로 구할 수 있음
- 이미지의 픽셀 값에 대한 경향성을 알 수 있으므로 시각적인 분석, 판단에 용이

```
void cv::calcHist ( const Mat *     images,
                    int             nimages,
                    const int *     channels,
                    InputArray      mask,
                    OutputArray     hist,
                    int             dims,
                    const int *     histSize,
                    const float **  ranges,
                    bool            uniform = true ,
                    bool            accumulate = false
                  )
```

images: 입력 영상의 주소(영상에 &을 붙여서 입력)

nimages: 입력 영상의 개수

channels: 히스토그램을 구할 채널을 나타내는 배열

mask: 계산을 수행할 범위 마스크(Mat()로 전체 선택)

hist: 히스토그램이 저장될 Mat

histSize: 가능한 모든 픽셀 레벨(bin이라 하며, 일반 8bit영상은 255)

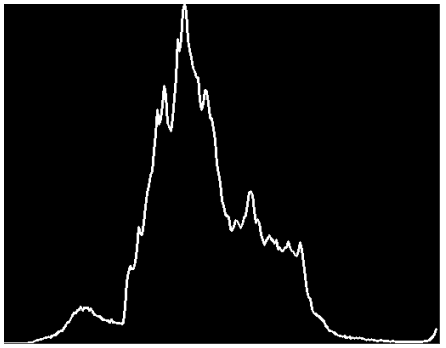
ranges: 히스토그램 범위

uniform: 히스토그램이 등간격인지 아닌지 선택

accumulate: 일반 히스토그램인지 누적 히스토그램인지 선택

Histogram

```
Mat GetHistogram(Mat& src) {
        Mat histogram;
        const int* channel_numbers = { 0 };
        float channel_range[] = { 0.0, 255.0 };
        const float* channel_ranges = channel_range;
        int number_bins = 255;

        // 히스토그램 계산
        calcHist(&src, 1, channel_numbers, Mat(), histogram, 1, &number_bins, &channel_ranges);

        // 히스토그램 plot
        int hist_w = 512;
        int hist_h = 400;
        int bin_w = cvRound((double)hist_w / number_bins);

        Mat histImage(hist_h, hist_w, CV_8UC1, Scalar(0, 0, 0));
        normalize(histogram, histogram, 0, histImage.rows, NORM_MINMAX, -1, Mat()); // 정규화

        for (int i = 1; i < number_bins; i++) {
                line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(histogram.at<float>(i - 1))),
                        Point(bin_w*(i), hist_h - cvRound(histogram.at<float>(i))),
                        Scalar(255, 0, 0), 2, 8, 0); // 값과 값을 잇는 선을 그리는 방식으로 plot
        }

        return histImage;
}
```



수행 결과

Operations

영상에 대한 산술 연산

- 산술 연산으로 두 영상을 합성하거나 영상의 값을 전체적으로 조절할 수 있음
- 산술 연산을 위한 함수가 따로 존재하지만 산술 연산자도 사용 가능

```
virtual void cv::MatOp::add ( const MatExpr &  expr1,
                              const MatExpr &  expr2,
                              MatExpr &        res
                            )                  const
virtual void cv::MatOp::add ( const MatExpr &  expr1,
                              const Scalar &   s,
                              MatExpr &        res
                            )                  const
virtual void cv::MatOp::subtract ( const MatExpr &  expr1,
                                   const MatExpr &  expr2,
                                   MatExpr &        res
                                 )                  const
virtual void cv::MatOp::subtract ( const Scalar &   s,
                                   const MatExpr &  expr,
                                   MatExpr &        res
                                 )                  const
```

```
Mat imgA = imread("landing.jpg", 1);
Mat imgB = imread("vin.png", 1);
resize(imgB, imgB, Size(imgA.cols, imgA.rows));
// 두 이미지 사이즈를 통일

Mat dst1, dst2, dst3, dst4;
add(imgA, imgB, dst1);
// dst1 = imgA + imgB;도 동일함
add(imgA, Scalar(100, 100, 100), dst2);
// dst2 = imgA + Scalar(100, 100, 100);도 동일함
subtract(imgA, imgB, dst3);
subtract(imgA, Scalar(100, 100, 100), dst4);
```

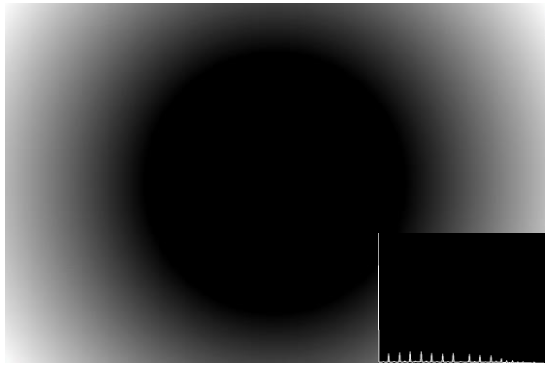
```
void cv::resize ( InputArray   src,
                  OutputArray  dst,
                  Size         dsize,
                  double       fx = 0,
                  double       fy = 0,
                  int          interpolation = INTER_LINEAR
                )
```

Operations

Img A

Img A + Img B

Img A + Scalar



Img B

Img A – Img B

Img A - Scalar

Logical Operations

- 영상의 논리 연산
  - 영상에 대한 논리적 연산으로 두 영상을 합성하거나 Masking할 수 있음

```
void cv::bitwise_and ( InputArray  src1,
                       InputArray  src2,
                       OutputArray dst,
                       InputArray  mask = noArray()
                     )
void cv::bitwise_or ( InputArray  src1,
                      InputArray  src2,
                      OutputArray dst,
                      InputArray  mask = noArray()
                    )
void cv::bitwise_not ( InputArray  src,
                       OutputArray dst,
                       InputArray  mask = noArray()
                     )
```

```
Mat image = imread("landing.jpg", 1);

Mat black(image.size(), CV_8UC3, Scalar(0));
// 동일한 크기의 검정 영상 생성
Mat mask(image.size(), CV_8U, Scalar(0));
// 동일한 크기의 마스크 생성
circle(mask, Point(500, 250), 200, Scalar(255), -1, -1);
// 마스크에 원 생성                                  채우기 플래그

Mat dst1, dst2;
bitwise_and(image, image, dst1, mask);
// 마스크 영역에서 image와 image의 AND연산
bitwise_or(black, image, dst2, mask);
// 마스크 영역에서 검정 영상과 image의 OR연산
```

```
void cv::circle ( InputOutputArray img,
                  Point            center,
                  int              radius,
                  const Scalar &   color,
                  int              thickness = 1 ,
                  int              lineType = LINE_8 ,
                  int              shift = 0
                )
```

Logical Operations

입력

AND 결과

=

OR 결과

Binarization

- 영상의 이진화
  - 영상을 두 개의 값으로 표현 하는 것 (0 or 1, 0 or 255 와 같이)
  - Grayscale영상에서 임계점보다 크고 작음을 기준으로 영상을 두 개의 값으로 나눌 수 있음
  - 적절한 임계 값을 설정하면 영상속에서 유용한 마스크를 획득할 수 있음
  - 적절한 임계 값은 히스토그램을 참조해 선택하거나 자동으로 찾아주는 알고리즘을 이용해 탐색(OTSU)

```
double cv::threshold ( InputArray  src,
                       OutputArray dst,
                       double      thresh,
                       double      maxval,
                       int         type
                     )
```

Parameters

src	input array (multiple-channel, 8-bit or 32-bit floating point).
dst	output array of the same size and type and the same number of channels as src.
thresh	threshold value.
maxval	maximum value to use with the **THRESH_BINARY** and **THRESH_BINARY_INV** thresholding types.
type	thresholding type (see **ThresholdTypes**).

THRESH_BINARY, THRESH_OTSU 등

Binarization

```
Mat color_img = imread("mark.jpg", 1);

Mat gray_img;
cvtColor(color_img, gray_img, CV_BGR2GRAY);
// 이진화를 위해 color에서 grayscale 영상으로 변환

Mat binary_img1, binary_img2;
threshold(gray_img, binary_img1, 127, 255, THRESH_BINARY);
// 임계값 지정 이진화
threshold(gray_img, binary_img2, 0, 255, THRESH_OTSU);
// 임계값 자동 이진화(OTSU 알고리즘)
```

입력 영상

THRESH_BINARY

THRESH_OTSU

Homework

실습 및 과제

- 주어진 영상(img1.jpg)에 빨강, 파랑, 초록 색의 점을 각각 설정한 개수만큼 무작위로 생성하는 프로그램을 작성할 것
- 앞서 생성한 영상에서 빨강, 파랑, 초록 색의 점을 각각 카운트하는 프로그램을 작성하고 카운트 결과가 실제와 일치하는지 검증할 것
- 주어진 영상을 이용해(img2.jpg) 다음과 같은 두 영상을 생성하는 프로그램을 작성하고(픽셀 값 접근을 이용) 히스토그램 일치 여부를 확인 및 그러한 결과가 나온 이유를 분석할 것

img2.jpg

위로 갈수록 점점 어두움

아래로 갈수록 점점 어두움

Homework

- 주어진 영상(img3.jpg, img4.jpg, img5.jpg)을 이용해 다음의 영상을 완성할 것

제출 파일

- 소스코드 파일 (반드시 .cpp, 또는 .h 파일만 첨부)
- 구현 방법 및 결과를 기술한 보고서 (워드 작성 후 PDF로 변환해 제출)