OPEN CV SEMINAR

Chap 1 · 2

2019/07/03

박 소 민

OPEN CV

CHAP 1 OpenCV 기초_ 영상처리와 비전

영상처리

영상을 처리하여
더 질 좋은 영상을 얻는 과정

- 영상 개선
- 영상 복원
- 영상 분할
- 영상 분류

컴퓨터 비젼

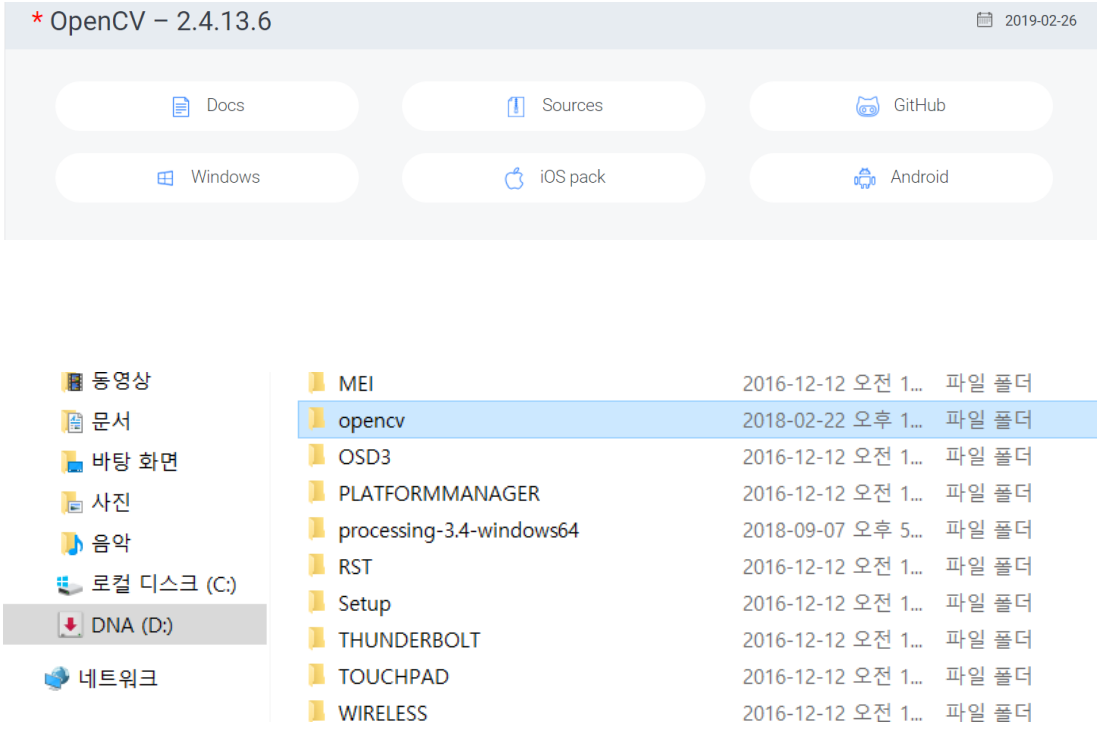
카메라에 의해 획득되는 영상에서
의미 있는 정보 추출

- 카메라 모델링
- 움직임/물체 검출 및 추적
- 스테리오 비전
- 3차원 물체 구조

Open CV 사용 영상처리, 비디오처리, 기계학습, 컴퓨터비전 라이브러리

CHAP 1 OpenCV 기초_ Open CV 라이브러리 설치

1단계 OpenCV 다운로드



https://opencv.org/releases/에서

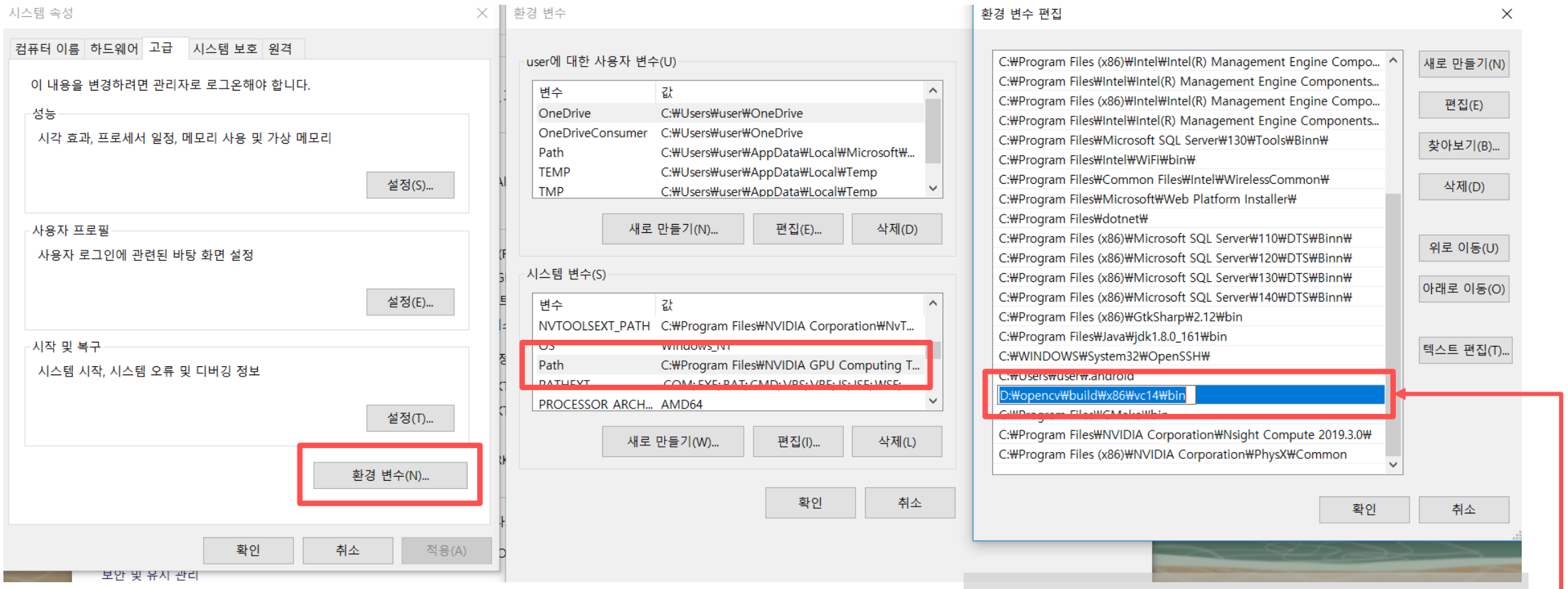
OpenCV-2.4.13.6 다운로드

D 드라이브에 압축 해제

“opencv” 파일 생성

CHAP 1 OpenCV 기초_ Open CV 라이브러리 설치

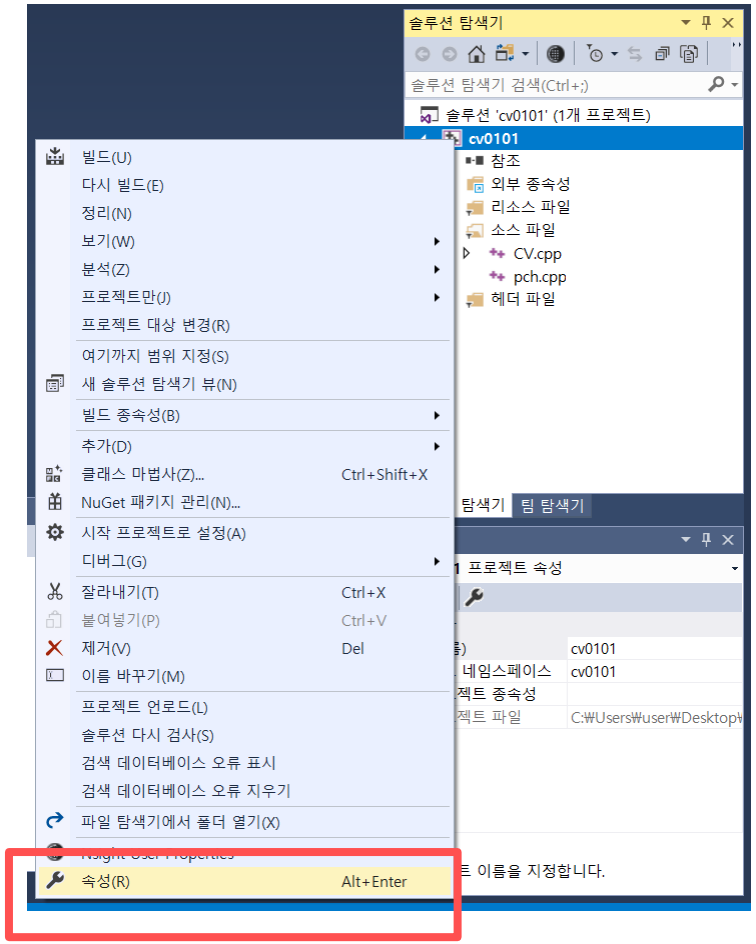
2단계 PATH 환경변수 설정 내 PC > 설정 > 고급 시스템 설정 > opencv bin 파일 주소 추가하기

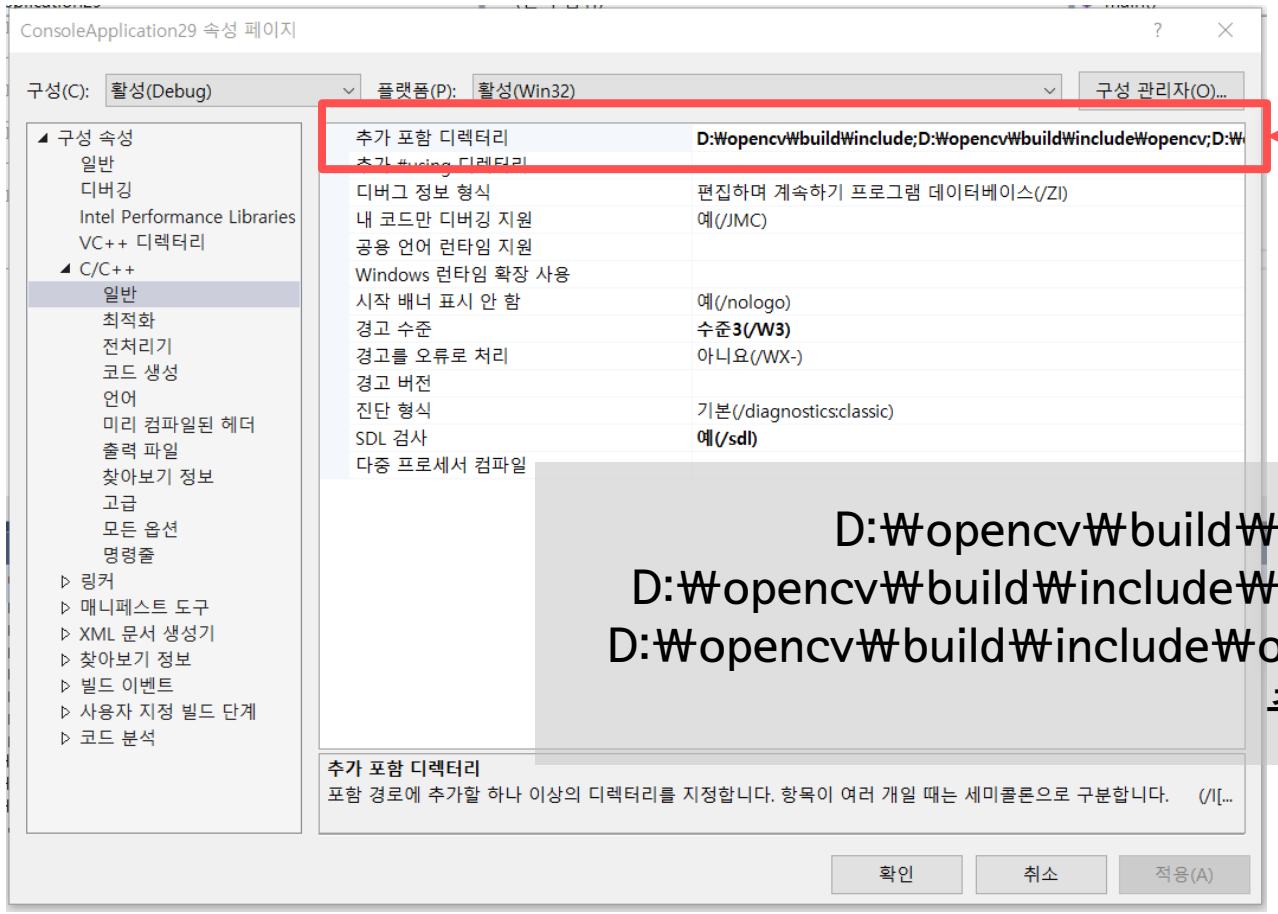


D:₩opencv₩build₩x86₩vc14₩bin

CHAP 1 OpenCV 기초_ Open CV 라이브러리 설치

3단계 Visual Studio 설정 — 추가 포함 디렉터리



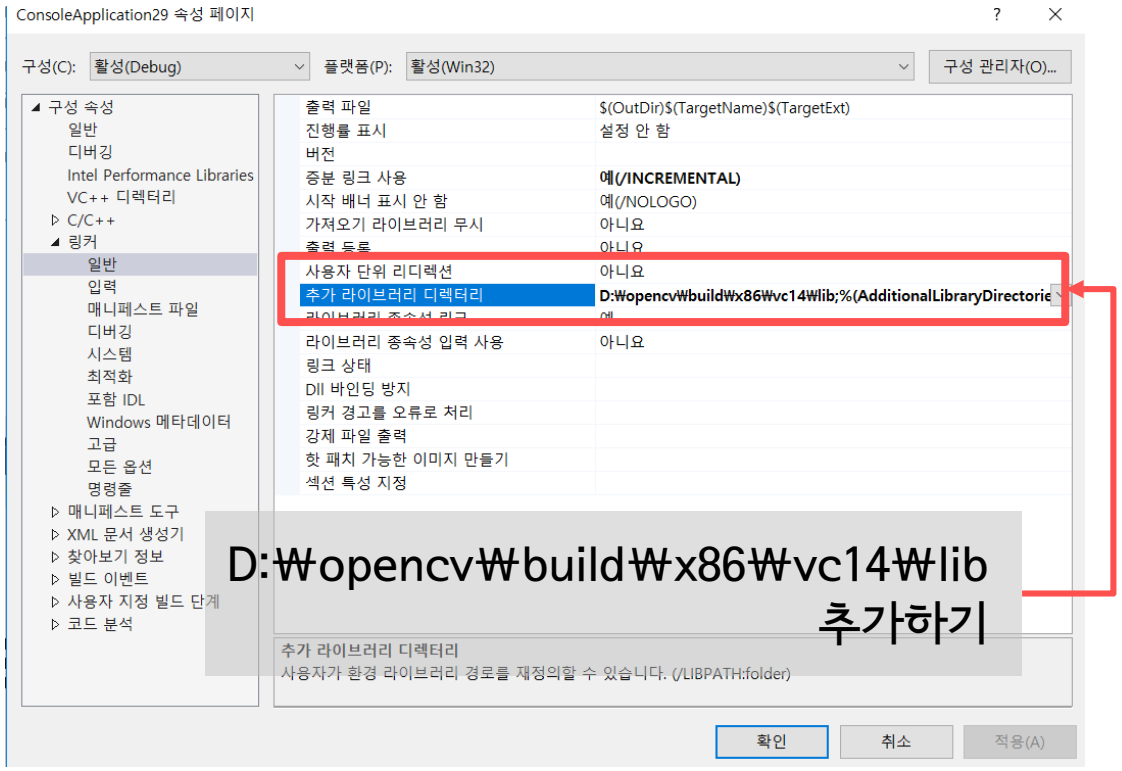


D:₩opencv₩build₩include
D:₩opencv₩build₩include₩opencv
D:₩opencv₩build₩include₩opencv2
추가하기

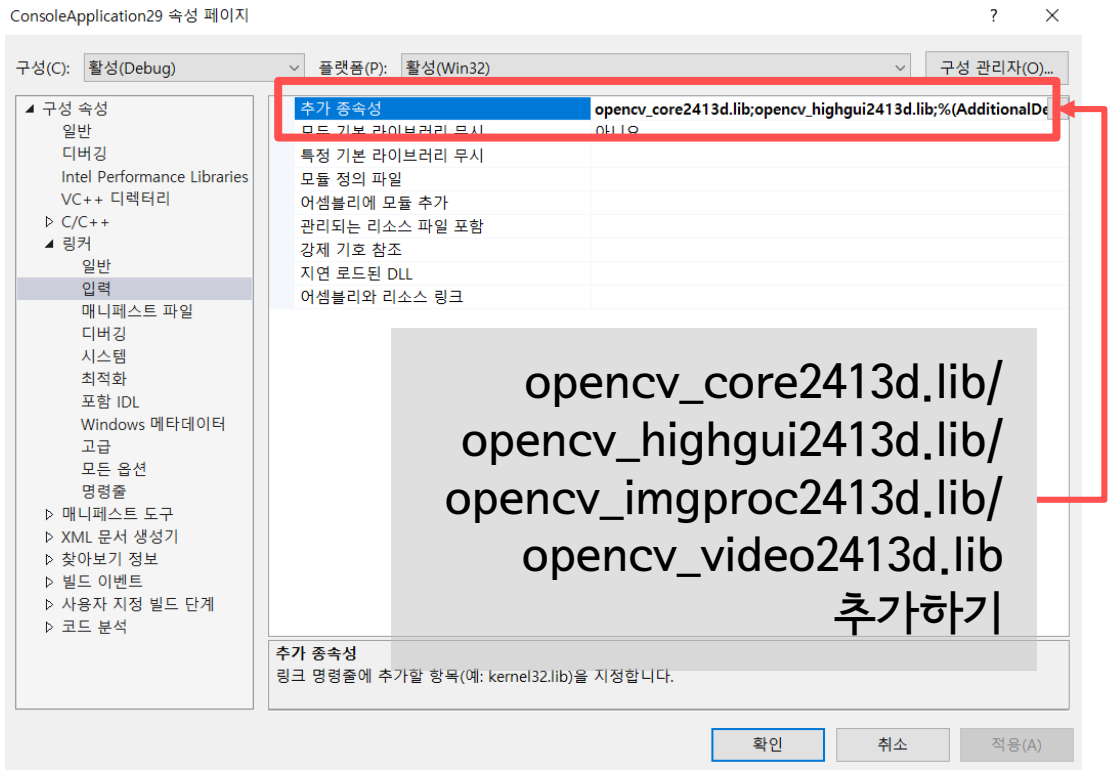
CHAP 1 OpenCV 기초_ Open CV 라이브러리 설치

3단계 Visual Studio 설정 추가 라이브러리 디렉터리, 추가 종속성

추가 라이브러리 디렉터리

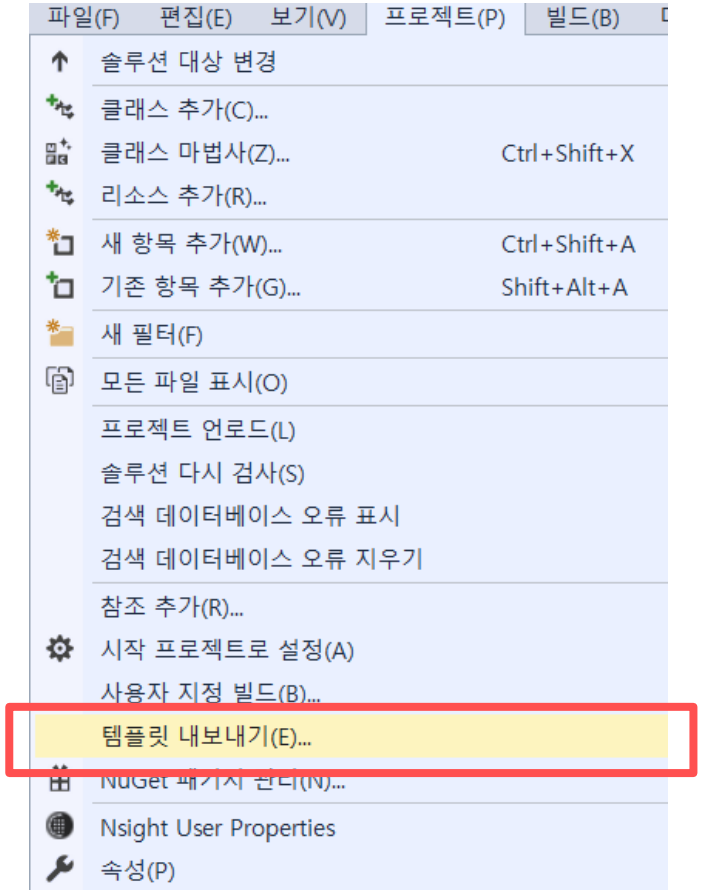


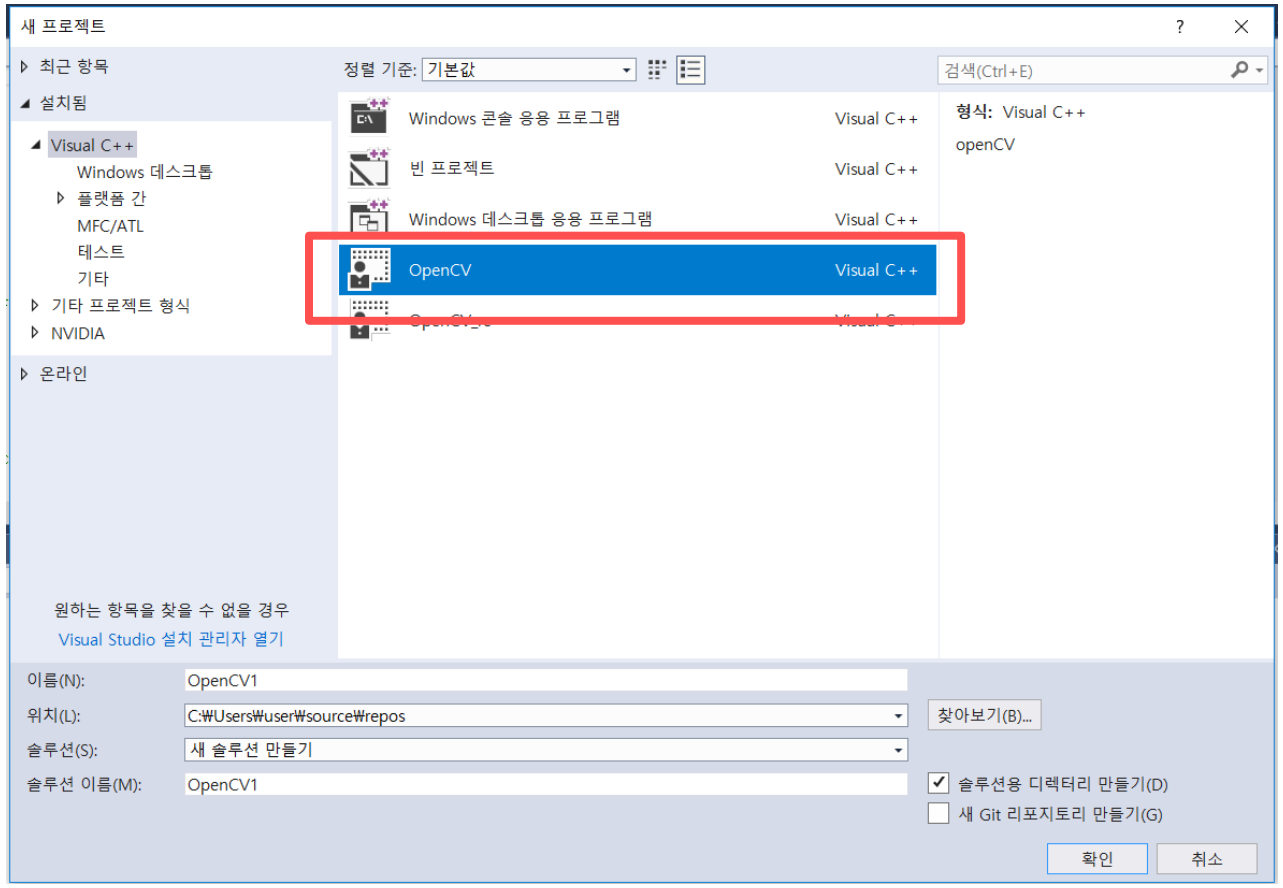
추가 종속성



CHAP 1 OpenCV 기초_ Open CV 라이브러리 설치

4단계 Visual Studio 2017 템플릿 저장





CHAP 2 OpenCV 기본 클래스 - 01

DataType

OpenCV의 기본 자료형을 표현하기 위한 템플릿 클래스

템플릿 클래스 등에서 **OpenCV 자료형으로 변환**하는 목적으로 사용

CV_<bit_depth>{U|S|F}C(<number_of_channels>)

MAX : 512, 1일 경우 생략 가능

Ex) CV_8UC3 8비트 깊이, uchar 3 채널 자료형

CV_32FC1 32비트 깊이, float 1 채널 자료형

Point_ / Point3_

Point_는 2D 좌표를, **Point3_**은 3D 좌표 템플릿 클래스

연산자나 dot(), ddot(), cross(), inside() 등의 메서드 사용가능

자료형 : Point, Point2i, Point2f, Point2d / Point3i, Point3f, Point3d

OPEN CV

CHAP 2 OpenCV 기본 클래스 - 01

Size_

크기를 표현하는 템플릿 클래스 연산자 사용가능

멤버변수 : **width**와 **height** 메서드 : **area()**

자료형 : Size, Size2i, Size2f

Rect_ / RotateRect

Rect_는 사각형을, **RotateRect**는 회전된 사각형을 표현하는 템플릿 클래스

Rect_ 자료형 : int

멤버변수 : (**x, y, width, height**) 메서드 : tl()topLeft, br()bottomRight, size(), area(), contains()

RotateRect

멤버변수 : ((**Point2f**)**center**, (**Size2f**)**size**, (**float**)**angle**) 메서드 : boundingRect()

회전된 사각형을 감싼 사각형

OPEN CV

CHAP 2 OpenCV 기본 클래스 - 02

Matx

고정된 작은 크기의 행렬 템플릿 클래스

float/double의 **1 * 1 ~ 6 * 6** 까지의 행렬

Ex) Matx23f float형의 2 *3 행렬

초기화

::zeros()

::ones()

::all(val)

::eye()

: 대각선이 1인 단위행렬

난수 생성

::randu(0.0, 1.0)

: 0~1 사이의 균등분포

::randn(0.0, 1.0)

: Ave 0, σ^2 1인 정규분포

행렬 연산

A.row(n)

A.col(n)

A.get_minor<n1,n2>(a,b)

A.mul(B) == A * B

A.dot(B)

A.t() : 전치행렬

A.reshape<n1,n2>

A.inv(DECOMP_CHOLESKY/LU)

: 역행렬

A.solve(B)

CHAP 2 OpenCV 기본 클래스 - 02

Vec

짧은 수치 벡터를 위한 템플릿 클래스(**Matx 상속**)

자료형 : Vec2b, Vec3b 등…

Ex) Vec<float, 3> X(1,0,0) == Vev3f X(1,0,0)

Vec<T,2>		Point_
Vec<T,3>	⟷	Point3_
Vec<T,4>		Scalar_, CvScalar

Scalar_

Vec 클래스에서 상속받은 **4개의 요소** 갖는 템플릿

Ex) Scalar_<uchar/int/float/double> S =
Scalar_<uchar/int/float/double>::all(255)

Scalar_ ⟷ CvScalar

cv.h 필요

CHAP 2 OpenCV 기본 클래스 - 02

Range

행 또는 열의 범위 지정하는 템플릿 클래스(Mat 클래스)

(n1, n2) : n1포함, n2 비포함

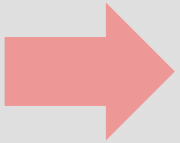
Ptr

포인터를 감싸서 **메모리를 안전하게 사용**하도록 하는 템플릿 클래스

Ex) int *ptr = new int[100];

Ptr<int> intData(ptr);

for(int I =0; i<100; i++) intData[i] = i;



메모리 누수(memory leak)가 일어나지 않음

동적 할당된 메모리의 해제를 파괴자에서

자동으로 실시함

CHAP 2 OpenCV 기본 클래스 - 03

Mat

*A.refcount 증가

1채널 혹은 다채널의 실수, 복소수, 행렬, 영상 등의 수치 데이터를 표현하는 n 차원 행렬 클래스

copyData false : 메모리 공유(기본) **true** : 복사하여 공유 안 함 (= Mat::clone())

size(3, 2) == 2 * 3 행렬

Ex) 2차원 : Mat A((rows, cols)/size, type) / (((rows, cols)/size, type, Scalar/*data) …

3차원 : Mat A(ndims, *sizes, type, Scalar/*data, …) …

Mat::create

2차원 : A.create(rows, cols, type) A.create(size, type)

다차원 : A.create(ndims, *sizes, type)

CHAP 2 OpenCV 기본 클래스 - 03

Mat 행렬 정보

Mat A(4, 5, CV_32FC3);

rows : 4

cols : 5

dims : 2

total() : 20

isContinuous() : 1 (각 행의 마지막에 연속으로 데이터가 저장되었는가?)

elemSize() : float(4byte) * 3(channels) = 12 byte

elemSize1() : elemSize()/channel = 4 byte

step() : cols * elemSize() = 60 byte

step1() : step()/rows = 15 byte

type() : CV_32FC3 = 21
depth() : CV_32F = 5

CHAP 2 OpenCV 기본 클래스 - 03

Mat 행렬 = 연산자 함수

Mat& Mat::operator = (const Mat& m)

오른쪽 행렬(m)이 왼쪽 행렬로 데이터를 복사하지 않고 공유됨

Mat& Mat::operator = (const MatExpr& expr)

오른쪽 행렬이 수식일 때 호출됨

Mat& Mat::operator = (const Scalar& s)

행렬의 모든 요소를 s 값으로 변경함

CHAP 2 OpenCV 기본 클래스 - 03

Mat::at

1 채널 행렬 접근

- A.at<float>(i, j)
- A.at<float>(Point(j, i))
- idx[0] = i; idx[1] = j;
  A.at<float>(idx)

n 채널 행렬 접근 → = <Vec<double,n>>

- A.at<Vec2d>(i, j)
- A.at<Vec3d>(i, j)
- A.at<Vec4d>(i, j) == A.at<Scalar>(i ,j)

Mat::ptr

1 채널 행렬 접근

```
Mat A(3,3,CV_32F);
float *ptrA = A.ptr<float>(i);
ptrA[j] = n;
```

i행 주소 저장

j열 요소 접근

3 채널 행렬 접근

```
Mat A(3,3,CV_32FC3);
ptrA[j] = Vec<float,3>(n1,n2,n3);
          or
ptrA[j*3], ptrA[j*3 + 1], ptrA[j*3 + 2]
```

```
Vec3f* ptrA = A.ptr<Vec3f>(i);
```

i행 주소 저장

j열 요소 접근

CHAP 2 OpenCV 기본 클래스 - 03

Mat 부분 행렬 헤더 → 행렬 데이터를 **공유**한다는 의미

.row(y)
.col(x)

.rowRange(startRow, endRow)
.colRange(startCol, endCol)
start ~ (end-1)까지

Mat 행렬의 복제, 복사, 변환, 값 설정 및 모양 변경

복제 .clone()

복사 .copyTo(OutputArray m)
행렬의 크기 및 자료형이 다르면 create()가 다시 생성

.copyTo(OutputArray m, InputArray mask)
mask에서 0이 아닌 값의 위치면 m으로 복사

변환 .convertTo(OutputArray m, int rtype, double alpha, double beta)
rtype으로 행렬 m을 변환
모든 행렬요소에 alpha를 곱하고, beta를 더하여 반환

.assignTo(Mat &m, int type)
alpha와 beta가 없는 convertTo()

값 설정 .setTo(InputArray val, InputArray mask)
행렬 요소를 val으로 설정

모양 변경 .reshape(int cn, int rows)

cn = 새로운 채널 개수 rows = 새로운 행의 개수, 0이면 유지

CHAP 2 OpenCV 기본 클래스 - 03

Mat 행렬의 메모리 해제, 크기 변경, 공간 확보

크기 변경

resize(size_t sz)
행의 개수를 sz로 변경

resize(size_t sz, const Scalar& s)
행의 개수를 sz로 변경,
s는 새로 추가된 요소의 값

메모리 해제

release()
행렬의 참조 카운터 1 감소
0이 되면 메모리 해제

공간 확보

reserve(size_t sz)
미리 메모리 용량 확보
재할당 빈번히 일어나는 것 방지

Mat 행렬의 ROI → Region Of Interest

locateROI(Size& wholeSize, Point& ofs)

부분 행렬에 의해 지정된 부분 행렬에서 원본 행렬의 전체 크기 wholeSize, 원본 행렬에서의 옵셋 위치 ofs 알려줌

adjustROI(int dtop, int dbottom, int dleft, dright)

ROI의 크기 및 위치 조정, 상대값!

CHAP 2 OpenCV 기본 클래스 - 03

Mat 행렬의 () 연산자 메서드

Mat Mat::operator() (Range rowRange, Range colRange) const

Mat Mat::operator() (const Rect& roi) const

Mat Mat:: Mat Mat::operator() (const Range* ranges) const

행렬의 부분 행렬의 헤더를 추출하는 연산자

CvMat(), IplImage()와 Mat 사이의 자료형 변환은 4장에서

Mat 행렬의 반복자

Mat::begin(), Mat::end()	행렬의 시작과 끝 요소로의 반복자 반환
MatIterator_	읽기와 쓰기 가능
MatConstIterator_	읽기만 가능

EX)

```
MatConstIterator_<float> it = A.begin<float>();
for(;it != A.end<float>();it++)
sum += *it;
```

CHAP 2 OpenCV 기본 클래스 - 03

Mat 행렬의 push_back, pop_back

Mat::push_back(const T& elem)

Mat::push_back(const Mat& m)

마지막에 elem/m을 추가

단, 모든 자료형이나 열의 개수가 일치해야함

Mat::pop_back(size_t nelems)

nelems 개구의 행 제거

Mat 행렬의 행렬 연산 메서드

::t() : 전치행렬

::inv(DECOMP_CHOLESKY/LU/SVD) : 역행렬

::mul(m, scale) : 두 행렬 요소간 곱셈!

::cross(m) : 외적

::dot(m) : 내적

::zeros(rows, cols, type)/(size, type)/(ndims, *sz, type)

::ones(rows, cols, type)/(size, type)

::eye(rows, cols, type)/(size, type) : 대각선이 1인 단위행렬

CHAP 2 OpenCV 기본 클래스 - 04

Mat_

Mat 클래스에서 상속된 템플릿 클래스, 메서드만 존재

Mat과 Mat_ 사이의 자료형 변환에 주의

행렬요소에 빈번히 접근하는 연산이나 행렬의 자료형이 미리 결정되면 Mat_이 더 편리

```
Mat A(2, 3, CV_32F);
Mat_<float> B = (Mat_<float>) A;
Mat_<float> &C = (Mat_<float>&) A;
```

레퍼런스 카운트 증가

동일한 주소
레퍼런스 카운트 유지

```
Mat_<float> D1(2, 3);
Mat_<float> D2(2, 3, 10);
Mat_<float> E1 = (Mat_<float>(2, 3) << 1,2,3,4,5,6);
Mat E2 = (Mat _<float>(2,3) <<1,2,3,4,5,6
```

CHAP 2 OpenCV 기본 클래스 - 04

Mat_ 행렬 생성

::create(int _rows, int _cols) ::create(Size _size) ::create(int _ndims, const int* _sizes)

Mat 행렬 생성과 매우 유사함, type 지정만 없음

type 부분은 기존 행렬과 동일하면서 크기가 다른 행렬을 생성할 때 사용

행렬의 크기가 다를 경우, 행렬 요소들을 저장할 메모리는 해제되었다가 재할당

Mat_ 행렬 정보

Mat_<Vec3f> A(4, 5, Vec3f(255,255,255));

type() : CV_32FC3 = 21 depth() : CV_32F = 5 channels() : 3

elemSize() : float(4byte) * 3(channels) = 12 byte

elemSize1() : elemSize()/channel = 4 byte

step() : cols * elemSize() = 60 byte

step1() : step()/rows = 15 byte

stepT() : step()/elemSize() = 5

CHAP 2 OpenCV 기본 클래스 - 04

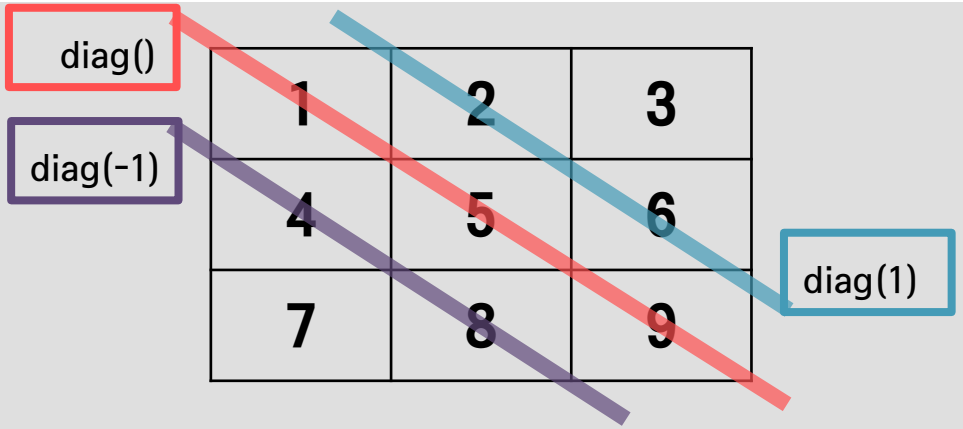
Mat_ 행렬의 행, 열, 대각선 지정에 의한 부분 행렬 헤더 생성 및 복제

::row(int y) const

::col(int x) const

::diag(int d) const

::clone() const



Mat_ 행렬의 ROI 조정 및 영역 관련 () 연산자 메서드

adjustROI(int dtop, int dbottom, int dleft, dright) ROI의 크기 및 위치 조정, 상대값!

Mat_ Mat_::operator() (const Range& rowRange, const Range& colRange) const

Mat_ Mat_::operator() (const Rect& roi) const

Mat_ Mat_::operator() (const Range* ranges) const

CHAP 2 OpenCV 기본 클래스 - 04

Mat_ 행렬의 지정 연산자 메서드 및 자료형 전환 () 연산자 메서드

Mat_& Mat_::operator = (const Mat& m) Mat_& Mat_::operator = (const Mat_& m)	Mat 클래스 객체와 Mat_ 클래스 객체를 =를 사용하여 연산자 왼쪽에 저장
Mat_& Mat_::operator = (const Scalar& s) Mat_& Mat_::operator = (const _Tp& s)	상수를 행렬의 모든 요소값으로 저장
Mat_& Mat_::operator = (const MatExpr& e)	상수를 행렬의 모든 요소값으로 저장

vector<_Tp>() 행으로 이루어진 행렬을 std::vector

Vec<typename DataType<_Tp>::channel_type, n>()

행 또는 열로 이루어진 행렬을 Vec으로 변환

Operator Matx<typename DataType<_Tp>::channel_type, m, n>

Matx 행렬로 변환

EX) Mat_<float>A(3, 4);

Vector<float> V1 = (vector<float>)A;

Vec<float,4> V2 = a;

Matx14f m1 = (Matx<float,1,4>) A;

CHAP 2 OpenCV 기본 클래스 - 04

Mat_ 행렬의 반복자

Mat_::begin(), Mat_::end()	행렬의 시작과 끝 요소로의 반복자 반환
Mat_::iterator	읽기와 쓰기 가능
Mat_::const_iterator	읽기만 가능

EX)

```
Mat_<float>::iterator it = A.begin();
for(;it != A.end();it++)        sum += *it;
```

CHAP 2 OpenCV 기본 클래스 - 05

InputArray, OutputArray 클래스

mat.hpp에 정의

OpenCV 함수, 클래스 메서드 등에서 **벡터나 행렬을 인수로 전달**할 때 사용

InputArray

OpenCV 함수의 **입력**으로 사용될 벡터나 행렬을 인수로 전달할 때 사용, 인수는 **읽기**만 가능

OutputArray

OpenCV 함수의 **출력**으로 사용될 벡터나 행렬을 인수로 전달할 때 사용, 인수는 **읽기와 쓰기** 가능

InputOutputArray

OutputArray에서 상속받은 클래스 OpenCV 함수의 **입출력**으로 사용될 벡터나 행렬을 인수로 전달할 때 사용

OPEN CV

CHAP 2 OpenCV 기본 클래스 - 06

vector 클래스

초기화

- vector<int> V1(3,0) : 크기 3, 0으로 초기화
- int arr[] = {1,2,3}
  vector<int> V2(arr, arr+sizeof(arr)/sizeof(arr[0])) : arr의 시작 주소~마지막 요소 주소를 사용하여 초기화
- vector<int> V3(V2.begin(), V2.end()) : 반복자 메서드를 이용한 초기화

배열로 사용

::push_back : 데이터 크기 지정하지 않고 데이터 추가

::size() : 저장된 데이터의 크기

::capacity() : 할당된 용량

::reserve() : 미리 메모리 용량 확보 (할당된 용량보다 크게 삽입이 일어나면 메모리 재할당 -> 성능저하)

CHAP 2 OpenCV 기본 클래스 - 06

vector 클래스

다른 자료형으로 변환

```
vector<Point3f> V1;
V1.push_back(Point3f(255,0,0));
V1.push_back(Point3f(0,255,0));
V1.push_back(Point3f(0,0,255));
V1.push_back(Point3f(0,255,255));

Mat A(V1);
```

```
vector<Point> contour;
contour.push_back(Point3f(100,100));
contour.push_back(Point3f(200,100));
contour.push_back(Point3f(200,200));
contour.push_back(Point3f(100,200));
```

- Point *P1 = &contour[0];
- Point *P2=(Point*)Mat(contour).data;
- Point P3[4];
  copy(contour.begin(), contour.end(), P3);
- 사용자 지정 템플릿 함수 사용

Pi[i]…로 접금

CHAP 2 OpenCV 기본 클래스 - 06

vector 클래스

다른 자료형으로 변환

```
vector<vector<Point>> contour(2,vector<Point>());
contour[0].push_back(Point3f(100,100)); contour[0].push_back(Point3f(200,100));
contour[0].push_back(Point3f(200,200)); contour[0].push_back(Point3f(100,200));
contour[1].push_back(Point3f(300,200)); contour[1].push_back(Point3f(400,100));
contour[1].push_back(Point3f(400,200));
```

- Point *pts1, *pts2; pts1 = &contour[0][0]; pts2 = &contour[1][0];
  Point *P1 = {pts1, pts2};
- pts1 =(Point*)Mat(contour[0]).data; pts2 =(Point*)Mat(contour[1]).data;
- Point P3[2][4];
  copy(contour[0].begin(), contour[0].end(), P3[0]);
  copy(contour[1].begin(), contour[1].end(), P3[1]);
- Point *ptrP3[] = {P3[0], P3[1]};
- 사용자 지정 템플릿 함수 사용

CHAP 2 OpenCV 기본 클래스 - 07

XML, YAML 파일 저장 및 읽기

```
//0258
FileStorage fs("test.xml", FileStorage::WRITE);

time_t date;
time(&date);
fs << "Date" << asctime(localtime(&date));

fs << "name" << "KDK";
fs << "age" << 25;

fs << "Images" << "[";
fs << "Apple.jpg" << "Banana.jpg"
<< "Orange.jpg";
fs << "]";

fs << "Box";
fs << "{" << "Left"<<100;
fs << "Top" << 200;
fs << "Right" << 300;
fs << "Bottom" << 400 << "}";

int arr[] = { 1,2,3,4,5,6,7, 8,9 };
vector<int> V1(arr, arr + sizeof(arr) / sizeof(arr[0]));
fs << "V1" << V1;

Point2f ptCenter(256.0f, 256.0f);
float angle = 45;
double scale = 10.0;
fs << "angle" << angle;
fs << "scale" << scale;
fs << "center" << ptCenter;

Mat matR = getRotationMatrix2D(ptCenter, angle, scale);
fs << "matR" << matR;
fs.release();
```

OPEN CV

CHAP 2 OpenCV 기본 클래스 - 07

XML, YAML 파일 저장 및 읽기

```
//0258
```


```
<?xml version="1.0"?>
- <opencv_storage>
    <Date>"Fri Jun 28 22:56:50 2019 "</Date>
    <name>KDK</name>
    <age>25</age>
    <Images> Apple.jpg Banana.jpg Orange.jpg</Images>
  - <Box>
      <Left>100</Left>
      <Top>200</Top>
      <Right>300</Right>
      <Bottom>400</Bottom>
    </Box>
    <V1> 1 2 3 4 5 6 7 8 9</V1>
    <angle>45.</angle>
    <scale>10.</scale>
    <center> 256. 256.</center>
  - <matR type_id="opencv-matrix">
      <rows>2</rows>
      <cols>3</cols>
      <dt>d</dt>
      <data> 7.071067811865475e+00 7.071067811865476e+00 -3.364386719675123e+03 -7.071067811865476e+00 7.071067811865475e+00 2.559999999999977e+02</data>
    </matR>
</opencv_storage>
```

```
    fs << "Bottom" << 400 << "}";
```

CHAP 2 OpenCV 기본 클래스 - 07

XML, YAML 파일 저장 및 읽기

```
//0259
FileStorage fs("test.xml", FileStorage::READ);

if (!fs.isOpened()) {
cerr << "The File is not oppend! FAIL" << endl;
return 1;
}
string date;
fs["Date"] >> date;
cout << "Date:" << date << endl;

string sName;
fs["name"] >> sName;
cout << "name:" << sName << endl;

int nAge;
fs["age"] >> nAge;
cout <<"age:" << nAge << endl;

float fAngle;
fs["angle"] >> fAngle;
cout << "angle:" << fAngle << endl;

double dScale;
fs["scale"] >> dScale;
cout << "scale:" << dScale << endl;

Point ptCenter;
fs["center"] >> ptCenter;
cout << "center:" << ptCenter << endl;

FileNode node = fs["Images"];
if (node.type() != FileNode::SEQ) {
cerr << "It is not a sequnce! FAIL" << endl;
return 1;
}

cout << "node[0]:"<<(string)node[0] << endl;
cout << "node[1]:" << (string)node[1] << endl;
cout << "node[2]:" << (string)node[2] << endl;

cout << node.name() << "=[";
FileNodeIterator it;
for (it = node.begin(); it != node.end(); it++) {
cout << (string)*it << ";";
}
cout << "]" << endl << endl;

node = fs["Box"];
if (node.type() != FileNode::MAP) {
cerr << "It is not a mapping! FAIL" << endl;
return 1;
}
cout << node.name() << "={";
cout << "Left:" << (int)(node["Left"]) << ";";
cout << "Top:" << (int)(node["Top"]) << ";";
cout << "Right:" << (int)(node["Right"]) << ";";
cout << "Bottom:" << (int)(node["Bottom"]) << ";";

vector<int> V1;
fs["V1"] >> V1;
cout << fs["V1"].name() << ":" << (Mat)V1 << endl
<< endl;
Mat matR;
fs["matR"] >> matR;
cout << fs["matR"].name() << ":" << matR << endl
<< endl;
fs.release();*/
```

CHAP 2 OpenCV 기본 클래스 - 07

XML, YAML 파일 저장 및 읽기

```
//0259
FileStorage fs("test

if (!fs.isOpened())
cerr << "The File is
return 1;
}
string date;
fs["Date"] >> date
cout << "Date:" <<

string sName;
fs["name"] >> sNa
cout << "name:" <

int nAge;
fs["age"] >> nAge;
cout <<"age:" << n

float fAngle;
fs["angle"] >> fAng
cout << "angle:" <
```

```
Microsoft Visual Studio 디버그 콘솔
Date:Fri Jun 28 22:56:50 2019

name:KDK
age:25
angle:45
scale:10
center:[256, 256]
node[0]:Apple.jpg
node[1]:Banana.jpg
node[2]:Orange.jpg
Images=[Apple.jpg;Banana.jpg;Orange.jpg;]

Box={Left:100;Top:200;Right:300;Bottom:400;V1:[1;
  2;
  3;
  4;
  5;
  6;
  7;
  8;
  9]

matR:[7.071067811865476, 7.071067811865475, -3364.386719675123;
  -7.071067811865475, 7.071067811865476, 255.9999999999998]


C:₩Users₩user₩Desktop₩opencv_PE₩02_vector₩Debug₩02_vector.exe(29012 프로세스)이(가) 0 코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버깅]->[디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도
록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요.
```

```
}
cout << "]" << endl << endl;
```

CHAP 2 OpenCV 기본 클래스 - 08

saturate_cast와 예외처리

uchar, schar, ushort, short , int, unsigned 자료형을 변환하면 변수 또는 연산 수식의 값이

자료형의 표현 범위에 있도록 보장

saturate_cast<int>는 float와 int 인수만을 입력으로 받음

데이터 형식, 값의 범위 등에 대한 예외 처리 가능

try에서 CV_Error, CV _Error_, CV_Assert 매크로를 이용하여 예외 발생시키고

catch 문장에서 cv::Exception 클래스로 처리 가능