

디지털 영상처리

영상처리, OpenCV와 파이썬

학습목표

- ▶ 컴퓨터 비전
- ▶ OpenCV와 파이썬 개요
- ▶ Visual code 설치

컴퓨터 비전

Preview

■ 놀라운 인간의 시각

- 인간은 아래 영상을 보고 인식, 추론, 예측, 상상 등을 수행함
- 선수가 얻을 점수까지 추정



그림 1-1 인간이 쉽고 정확하게 해석할 수 있는 영상

■ 컴퓨터가 인간 시각을 흉내 낼 수 있을까?

1.1 인간의 시각

■ 인간 시각의 강점

- 분류, 검출, 분할, 추적, 행동 분석에 능숙
- 3차원 복원 능력
- 빠르고 강건
- 다른 지능 요소인 지식 표현, 추론, 계획과 협동
- 사전 행동_{proactive}에 능숙
- 과업 전환이 매끄럽고 유기적이고 빠름
- 비주얼 서보잉이 뛰어남

1.1 인간의 시각

■ 인간 시각의 한계

- 착시가 있음

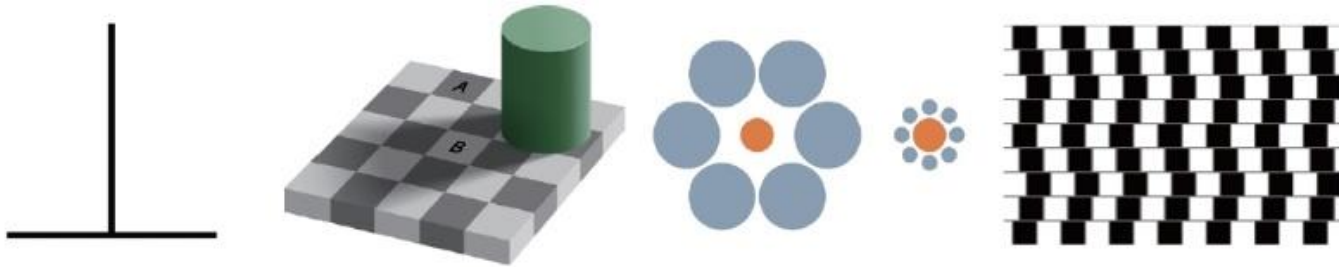


그림 1-3 인간 시각의 착시 현상(출처: 영문 위키피디아 'optical illusion')

- 정밀 측정에 오차
- 시야가 한정됨
- 피로해지고 퇴화

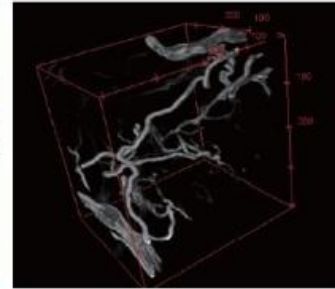
1.2 왜 컴퓨터 비전인가?

■ 몇 가지 대표적인 응용 사례

- 농업
- 의료
- 교통
- 스마트 공장
- 스포츠
- 유통



(a) 과일 수확 드론



(b) 혈관 분할



(c) 자율주행



(d) 불량 검사



(e) 선수의 행동 분석



(f) 고객의 동선 분석

1.3 컴퓨터 비전은 왜 어려운가?

■ 컴퓨터 비전이 어려운 이유는 명확

■ 세상의 변화무쌍함

- 환경 (낮밤, 날씨 등) 변화, 보는 위치와 방향의 변화, 강체와 연성 물체
- 원자부터 우주까지 긴 스펙트럼에서 영상 수집

■ 컴퓨터는 넘버 크런처

125	134	125	122	127	127	120	130	139	135	139	140	133	127	127	130	133	135	138	133	137	139	134	130	125	121
117	123	114	116	120	122	118	120	122	117	122	126	124	117	106	100	99	102	105	120	118	113	109	105	106	111
109	110	105	102	112	123	130	135	147	171	191	184	183	174	157	139	124	107	90	92	87	88	92	93	88	89
108	105	100	116	117	129	163	195	210	217	205	215	211	198	185	176	167	143	117	91	80	77	88	91	84	79
107	103	102	120	146	173	200	193	172	165	138	141	135	123	118	125	139	143	137	121	99	94	85	88	82	81
104	107	115	134	159	171	170	136	115	129	107	83	83	82	80	83	90	103	113	125	108	93	91	90	86	83
107	120	137	160	150	125	139	150	167	174	115	99	94	93	98	98	89	87	91	104	103	99	97	95	94	95
111	133	156	134	151	157	189	206	216	212	136	114	92	83	97	110	108	100	98	97	101	101	95	92	103	120
130	145	164	165	185	213	219	210	212	196	158	108	123	137	137	123	111	121	134	145	132	130	147	159	163	171
138	151	170	185	195	215	222	211	214	218	209	160	152	151	157	163	166	167	166	159	155	160	180	193	195	193
142	153	171	190	190	204	218	213	207	214	218	213	204	195	192	189	183	178	173	161	159	163	171	183	189	187
141	151	164	188	178	180	197	204	201	197	196	196	193	190	187	176	163	157	156	156	161	163	166	174	186	192
144	151	160	185	183	176	176	187	192	191	188	193	184	178	177	174	165	156	151	148	163	177	182	188	200	203
152	160	168	176	193	193	182	180	180	174	172	164	161	159	154	146	140	143	149	173	184	190	190	193	199	205
159	168	178	178	202	206	197	194	187	175	175	167	172	179	180	176	176	188	203	215	212	206	204	202	204	205
161	171	185	197	210	204	199	211	210	208	212	219	210	206	215	225	228	220	215	214	209	210	214	216	211	200



그림 1-6 컴퓨터 비전이 인식해야 하는 영상은 아주 큰 숫자 배열

■ 인공지능의 미숙함

- 지식 표현, 추론, 계획, 학습이 유기적으로 동작할 때만 강한 인공지능 가능
- 강한 인공지능은 먼 미래의 일 또는 영영 불가능

1.4 컴퓨터 비전의 역사

■ 신문 산업에서 태동한 디지털 영상

- 1920년 유럽과 북미 간 케이블을 통해 사진 전송하는 Bartlane 시스템 개통

■ 1946년 세계 최초의 범용 전자식 컴퓨터인 에니악 탄생

- 빠른 계산이 주목적 (에니악은 초당 3000개 가량 덧셈)

■ 1957년 스캐너를 통해 디지털 영상을 컴퓨터에 저장

- 5cmx5cm 사진에서 획득한 176x176 디지털 영상 ← 컴퓨터 비전의 태동



Bartlane 시스템(유럽-북미 해저 케이블로 사진을 전송하는 시스템)으로 전송된 세계 최초의 디지털 영상(1920년)

→
37년



세계 최초로 컴퓨터에 저장된 디지털 영상 (1957년)

→
65년



자율주행차(현재)

그림 1-7 컴퓨터 비전의 발전

1.4 컴퓨터 비전의 역사

표 1-1 컴퓨터 비전의 역사

연도	사건
1920	• Bartlane 영상 전송 케이블 시스템 구축 [McFarlane1972]
1946	• 세계 최초 전자식 범용 디지털 컴퓨터인 에니악 탄생
1957	• 커쉬가 세계 최초로 디지털 영상을 컴퓨터에 저장
1958	• 로젠블라트의 퍼셉트론 제안(이후 Mark 1 Perceptron에서 문자 인식 실험)
1968	• 소벨의 소벨 에지 연산자 제안
1979	• IEEE Transactions on Pattern Analysis and Machine Intelligence 창간 • ACRONYM 시스템 발표 [Brooks1979]
1980	• 후쿠시마의 네오코그니트론 논문 발표 [Fukushima1980]
1983	• 제1회 CVPR(Computer Vision and Pattern Recognition)이 미국 알링턴에서 개최
1986	• 캐니의 캐니 에지 연산자 논문 발표 [Canny1986] • 루멜하트의 『Parallel Distributed Processing』 출간(다층 퍼셉트론 제안) [Rumelhart1986]

1.4 컴퓨터 비전의 역사

1987	<ul style="list-style-type: none">• International Journal of Computer Vision 창간• 런던에서 제1회 ICCV(International Conference on Computer Vision) 개최(홀수 연도)• Marr상 제정(ICCV에서 시상)• 덴버에서 제1회 NIPS(Neural Information Processing Systems) 개최(2018년에 NeurIPS로 개명)
1990	<ul style="list-style-type: none">• 프랑스 안티베에서 제1회 ECCV(European Conference on Computer Vision) 개최(짝수 연도)
1991	<ul style="list-style-type: none">• Eigenface 얼굴 인식 논문 발표 [Turk1991]
1998	<ul style="list-style-type: none">• 르쿤의 컨볼루션 신경망 논문 발표 [LeCun1998]
1999	<ul style="list-style-type: none">• 로우의 SIFT 논문 발표 [Lowe1999]• 엔비디아에서 GPU 발표
2000	<ul style="list-style-type: none">• CVPR에서 OpenCV 알파 버전 공개
2001	<ul style="list-style-type: none">• Viola-Jones 물체 검출 논문 발표 [Viola2001]
2004	<ul style="list-style-type: none">• 그랜드 챌린지(고속도로 자율주행)
2005	<ul style="list-style-type: none">• PASCAL VOC 대회 시작

1.4 컴퓨터 비전의 역사

2006	<ul style="list-style-type: none">• OpenCV 1.0 공개
2007	<ul style="list-style-type: none">• 어번 챌린지(도심 자율주행)• Azriel Rosenfeld Lifetime Achievement상 제정
2009	<ul style="list-style-type: none">• 페이페이 리가 CVPR에서 ImageNet 데이터셋 발표• OpenCV 2.0 공개
2010	<ul style="list-style-type: none">• Xbox 360을 위한 Kinect 카메라 시판• 제1회 ILSVRC 대회 개최• MS COCO 데이터셋 발표
2012	<ul style="list-style-type: none">• ILSVRC 대회에서 AlexNet 우승 [Krizhevsky2012]• 시각 장애인을 태운 자율주행차의 시범 운행 성공
2013	<ul style="list-style-type: none">• 아타리 비디오 게임에서 사람 성능 추월 [Mnih2013]• 스코츠데일에서 제1회 ICLR(International Conference on Learning Representations) 개최
2014	<ul style="list-style-type: none">• RCNN 논문 발표 [Girshick2014]• 생성 모델인 GAN 발표 [Goodfellow2014]• ILSVRC에서 GoogLeNet이 우승, VGGNet이 준우승
2015	<ul style="list-style-type: none">• 텐서플로 서비스 시작• ILSVRC에서 ResNet이 우승

1.4 컴퓨터 비전의 역사

2016	<ul style="list-style-type: none">• 파이토치 서비스 시작• YOLO 논문 발표 [Redmon2016]
2017	<ul style="list-style-type: none">• 트랜스포머 논문 발표 [Vaswani2017]• Open Images 데이터셋 공개• 구글 렌즈 서비스 시작
2018	<ul style="list-style-type: none">• 인공지능이 그린 에드몽 벨라미가 경매에서 5억 원에 낙찰• 벤지오, 힌튼, 르쿤 교수가 딥러닝으로 튜링상 수상
2019	<ul style="list-style-type: none">• 알파스타가 스타크래프트에서 그랜드마스터 수준 달성• 트랜스포머를 위한 파이썬 라이브러리 transformers 2.0 공개
2020	<ul style="list-style-type: none">• OpenAI 재단의 GPT-3 발표• iPad Pro에 라이다 센서 장착
2021	<ul style="list-style-type: none">• 비전 트랜스포머 발표 [Dosovitskiy2021]• OpenAI 재단의 DALL·E 발표 [Ramesh2021]
2022	<ul style="list-style-type: none">• 구글의 Imagen 발표 [Saharia2022]

1.5 컴퓨터 비전 체험 서비스

■ 컴퓨터 비전 커뮤니티의 공개 문화

- SOTA 달성한 연구자는 논문 발표와 더불어 깃허브에 소스 코드와 데이터 공개하는 문화
- 이를 활용한 웹/앱 서비스 활성화



그림 1-8 Google 앱

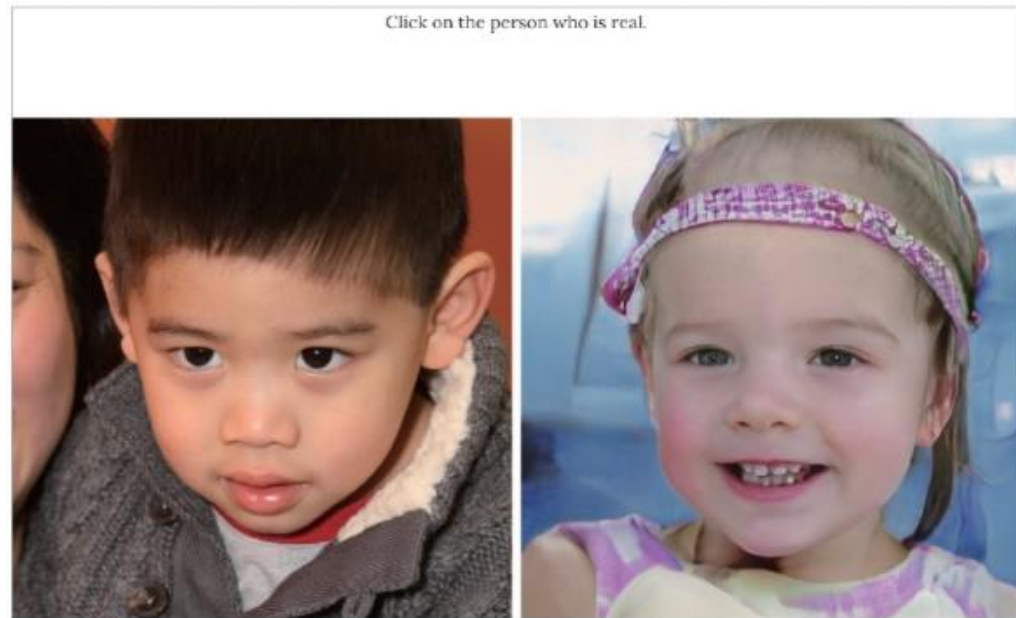


그림 1-9 Which face is real?(왼쪽이 진짜)

1.5 컴퓨터 비전 체험 서비스

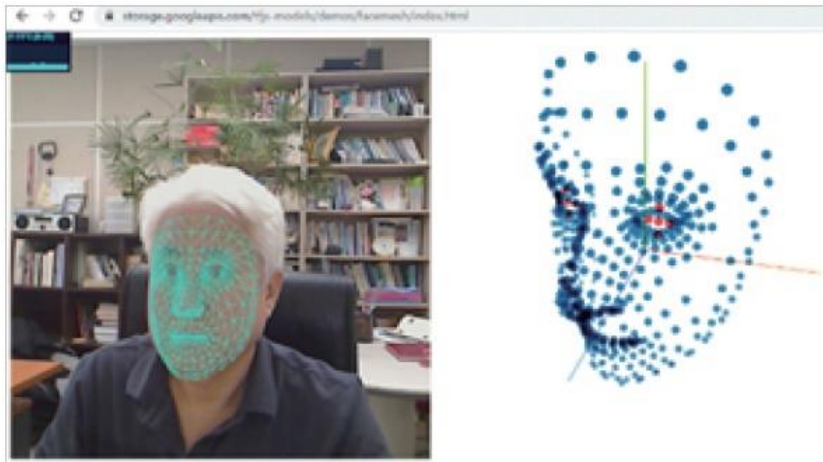


그림 1-10 얼굴 랜드마크 검출



그림 1-11 영상 설명

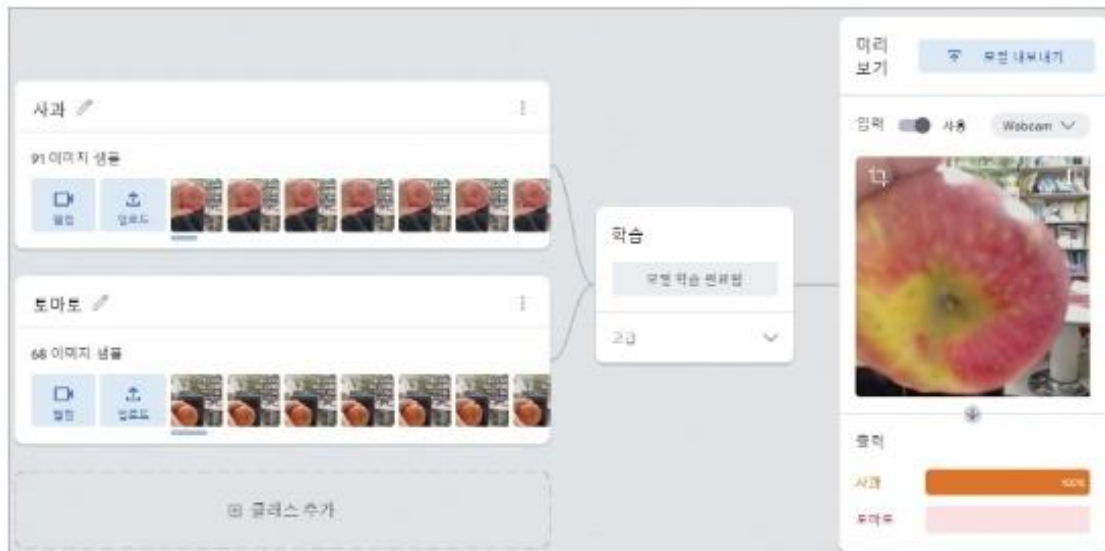


그림 1-12 티처블 머신

1.6 컴퓨터 비전 만들기

■ 궁극적인 목표

- 일반적인 상황에서 잘 작동하는 인간과 같은 시각 (강한 인공지능)
- 영영 불가능하거나 먼 미래에 실현

■ 현실적인 목표

- 제한된 환경에서 특정 과업을 높은 성능으로 달성 (약한 인공지능)
- 컴퓨터 비전 문제를 여러 세부 문제로 구분하고 세부 문제별로 알고리즘 구상

컴퓨터 비전이 풀어야 할 문제

■ 기본 문제

- 분류
- 검출
- 분할
- 추적
- 행동 분석
- ...



(a) 분류



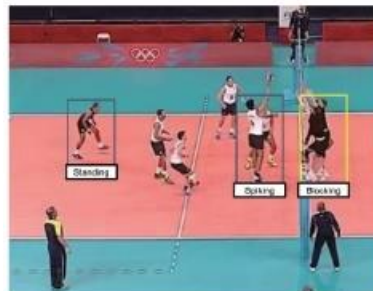
(b) 검출



(c) 분할



(d) 추적(<https://motchallenge.net/vis/MOT17-09-SDP>)



(e) 행동 분석(<https://github.com/mostafa-saad/deep-activity-rec#dataset>)

그림 1-13 컴퓨터 비전이 풀어야 할 문제

컴퓨터 비전이 풀어야 할 문제

■ 특정 상황에 따라 다양하게 변형

- 예) 사과 따는 로봇 비전 → 사과 검출에만 집중. 로봇 손을 위해 정확한 위치가 중요

■ 다른 지능 요소와 협업

- 가장 활발한 협업 분야는 자연어 처리, 예) 영상 설명하기
- 로봇과 협업은 활발, 예) 비주얼 서보잉을 통한 눈-손 협업

컴퓨터 비전 알고리즘과 프로그래밍

■ 고전 컴퓨터 비전과 딥러닝 컴퓨터 비전

- 대략 2010년을 기점으로 방법론의 대전환. 고전 방법은 규칙 기반, 딥러닝은 데이터 중심
- 둘 다 이해하는 것이 중요

■ 프로그래밍 언어

- C/C++, 자바, 파이썬을 주로 사용. 이 책은 파이썬을 사용
- 주로 C/C++는 알고리즘 구현, 파이썬은 인터페이스 언어로 활용됨
- 컴퓨터 비전 지원하는 OpenCV와 딥러닝 지원하는 텐서플로 및 파이토치 라이브러리 사용

■ 파이썬

- 배열 처리에 유리
- 파이썬 기초를 다지고 이 책을 공부해야 함

[C 언어]

```
for(i=0; i<n; i++) z[i]=x[i]+y[i];
```

[파이썬 언어]

```
z=x+y
```

Preview

■ 현대는 양호한 프로그래밍 환경

- 예전에는 알고리즘을 바닥부터 구현
- 현대는 함수 호출로 영상 처리하는 시대. 대표적 컴퓨터 비전 라이브러리는 OpenCV

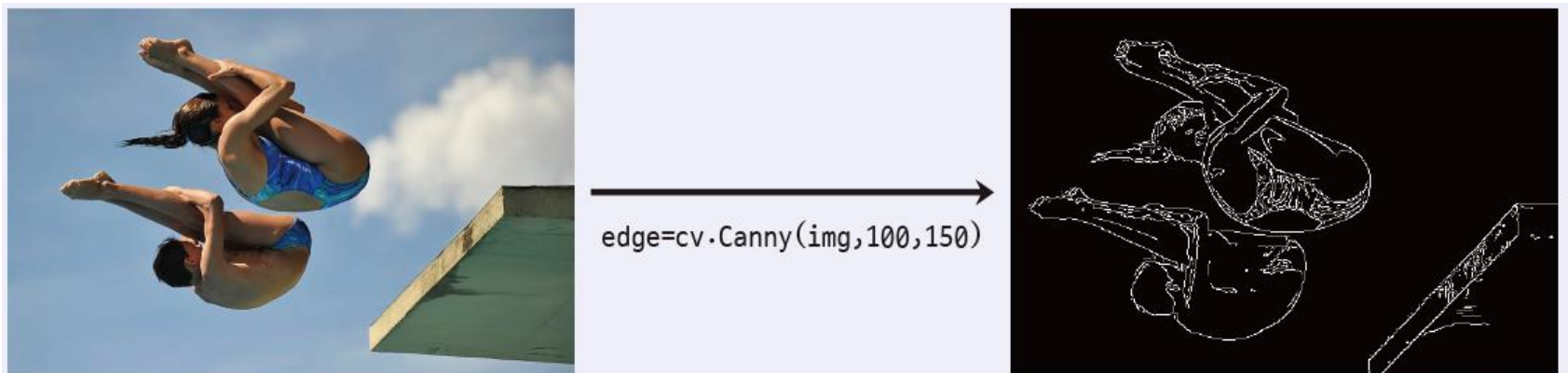


그림 2-1 에지 검출이라는 큰 일을 거뜬히 수행하는 OpenCV의 Canny 함수

■ 파이썬 언어와 OpenCV 라이브러리로 컴퓨터 비전 세계를 활짝~~~

2.1 OpenCV 소개

■ 인텔이 만들어 공개한 OpenCV

- 바퀴를 다시 발명 reinventing the wheel 하는 쓸데없는 노력을 방지할 목적
- 인텔 칩의 성능을 평가할 목적

개요와 간략한 역사

■ 개요

- 클래스와 함수는 C와 C++로 개발. 전체 코드는 180만 라인 이상
- 인터페이스 언어는 C, C++, 자바, 자바스크립트, 파이썬
- OS 플랫폼은 윈도우, 리눅스, macOS, 안드로이드, iOS
- 교차 플랫폼 지원
- 교육과 상업 목적 모두 무료

■ OpenCV - Open Source Computer Vision Library

- 영상 처리와 컴퓨터 비전 관련 오픈 소스 라이브러리
- 2,500개가 넘는 알고리즘으로 구성
 - 영상 처리, 컴퓨터 비전, 기계 학습과 관련된 전통적인 알고리즘
 - 얼굴 검출과 인식, 객체 인식, 객체 3D 모델 추출, 스테레오 카메라에서 3D 좌표 생성
 - 고해상도 영상 생성을 위한 이미지 스티칭, 영상 검색, 적목 현상 제거, 안구 운동 추적
 - 4만 7천 이상의 사용자 그룹과 1,800만 번 이상의 다운로드 횟수
- 구글, 야후, 마이크로소프트, 인텔, IBM, 소니, 혼다, 도요다와 같은 대기업부터 Applied Minds, Videosurf 및 Zeitera와 같은 신생 기업들까지 사용
- C, C++, 파이썬(Python), Java, 매트랩 인터페이스 제공
- 윈도우즈, 리눅스, 안드로이드, 맥 OS 등 다양한 운영체제 지원
- MX(Multimedia Extension)와 SSE(streaming SIMD Extensions) 명령어 통해 고속의 알고리즘 구현
- CUDA와 OpenCL 인터페이스 개발

〈표 2.1.1〉 OpneCV 버전별 특징

1.0 버전	2.0 버전	2.2 버전
<ul style="list-style-type: none"> • C 언어 기반 API • 구조체 기반 데이터 구조 사용 • 비주얼 스튜디오에서 라이브러리 컴파일 후 사용 • highgui 모듈에서 8비트 PNG, JPEG2000 입출력 지원 • 샘플 예제 파일 추가 (calibrate.cpp, inpaint.cpp, leter_recog. cpp 등) 	<ul style="list-style-type: none"> • C++ 언어 기반 API • 클래스 기반 데이터 구조 도입 • CMake를 이용하여 라이브러리 컴파일 후 사용 가능 • highgui 모듈에서 스테레오 카메라 지원 • 소스 디렉터리 구조 구성 	<ul style="list-style-type: none"> • 템플릿 자료구조 추가 • 기존 5개 라이브러리를 12개의 모듈로 재구성(opencv_core, opencv_imgproc, opencv_highgui, opencv_ml 등) • 안드로이드 지원 가능 • highgui 모듈에서 16비트 LZW-압축 TIFF 지원 • GPU 처리 지원
2.4 버전	3.0 버전	3.4 버전
<ul style="list-style-type: none"> • cv::Algorithm 클래스 도입 • SIFT와 SURF 모듈 유료화 • SIFT 성능 대폭 개선 • 컬러 영상 캐니 에지 수행 	<ul style="list-style-type: none"> • cv::Algorithm 적극 사용 • 1500개 패치 github 제출 • OpenCL을 사용하는 투명 GPU 가속 레이어 도입 • NEON 내장 함수 사용한 OpenCV 함수 가속화 • Python & Java 바인딩 확장 및 Matlab 바인딩 도입 • Python 3.0 지원 향상 • 안드로이드 지원 향상 • 비디오 캡처 및 멀티스레딩 함수 개선 	<ul style="list-style-type: none"> • dnn 모듈 개선 <ul style="list-style-type: none"> - fast R-CNN 지원 - Javascript 바인딩 - OpenCL 가속화 포함 • OpenCL 커널 바이너리에 디스크 캐시 및 수동 로딩 구현 • GSoC 프로젝트 통합으로 백그라운드 감산 알고리즘 구현

OpenCV

4.0 버전	4.1 버전	4.2 버전
<ul style="list-style-type: none">• 1.x 버전 C API 대량 제거• 효과적인 그래픽 기반 영상처리 엔진으로 G-API 모듈 추가• OpenVION 딥러닝 툴킷으로 dnn 모듈 업데이트• 키넥트 퓨전 알고리즘 구현• QR코드 검출기 추가• 효과적인 광류 알고리즘 추가	<ul style="list-style-type: none">• core와 imgproc 모듈 실행 최적화• dnn 모듈 개선<ul style="list-style-type: none">- NN Builder API로 교체- 인텔 Neural ComputerStick2 지원• 안드로이드 미디어 NDK API 지원• Hand-Eye 캘리브레이션 추가	<ul style="list-style-type: none">• dnn 모듈 개선<ul style="list-style-type: none">- cuda와 통합된 GSoC 프로젝트• 성능 개선<ul style="list-style-type: none">- SIMD 지원 확대- pryDown 멀티스레딩 지원• FSR 알고리즘

공식 사이트

■ OpenCV를 지원하는 사이트

- 공식 홈페이지(<https://opencv.org>)
- 매뉴얼 사이트: 프로그래밍할 때 가장 많은 도움(<https://docs.opencv.org>)
... 다음 슬라이드
- 깃허브
- 대한민국 OpenCV 사이트(<https://cafe.naver.com/opencv>)

공식 사이트

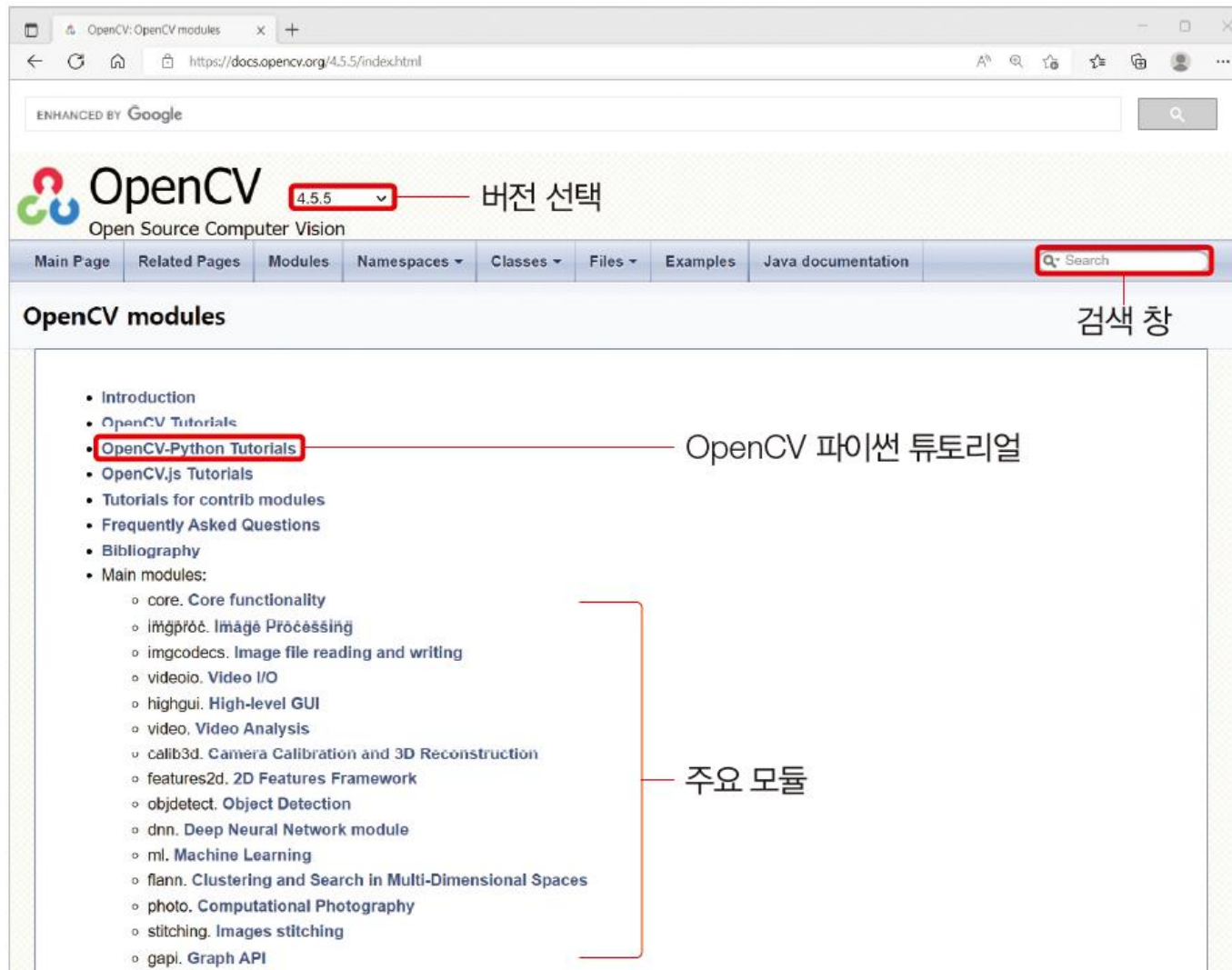


그림 2-2 OpenCV 매뉴얼 사이트

Visual Code 설치

MS Visual Studio Code

 Visual Studio Code Docs Updates Blog API Extensions FAQ Learn  Search Docs  Download

Version 1.71 is now available! Read about the new features and fixes from August.

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

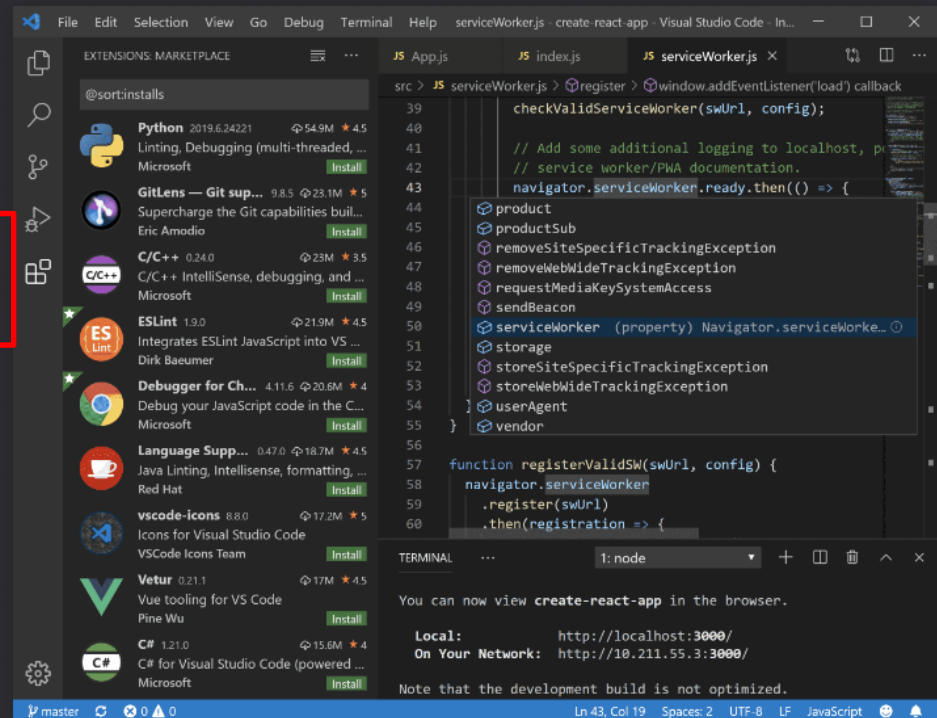
Download for Windows

Stable Build



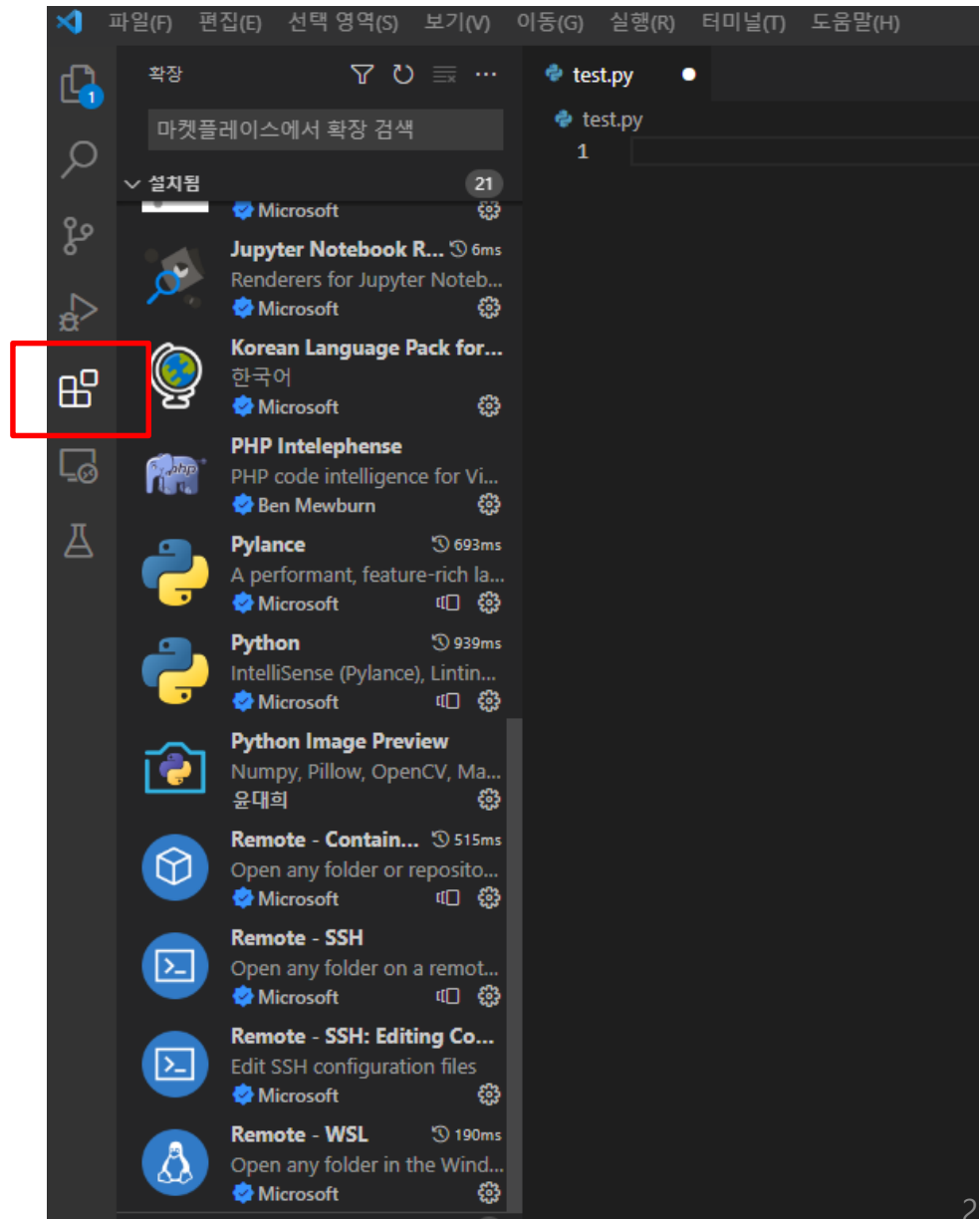
Web, Insiders edition, or other platforms

By using VS Code, you agree to its
[license](#) and [privacy statement](#).



Visual Code 설치

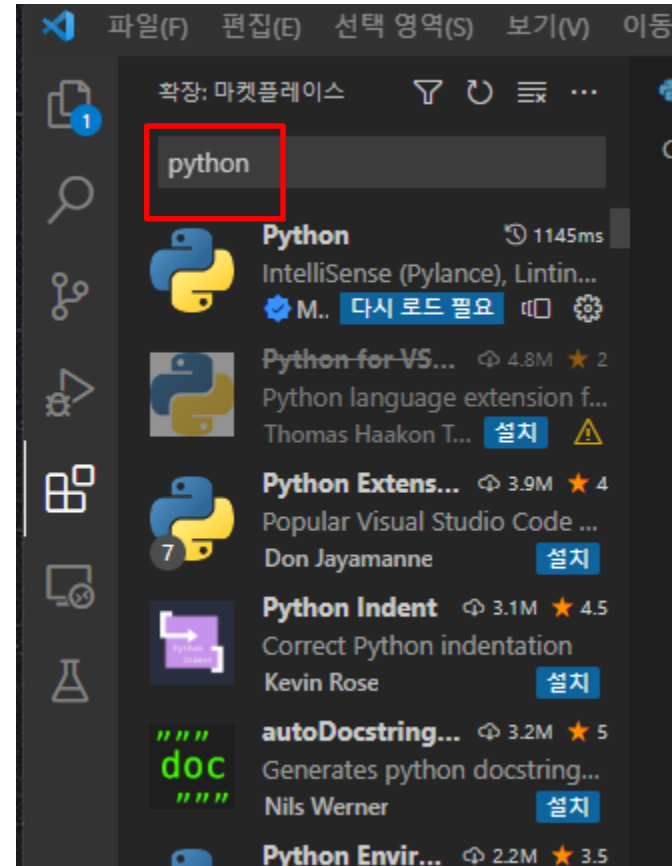
■ 파이썬 설치



Visual Code 설치

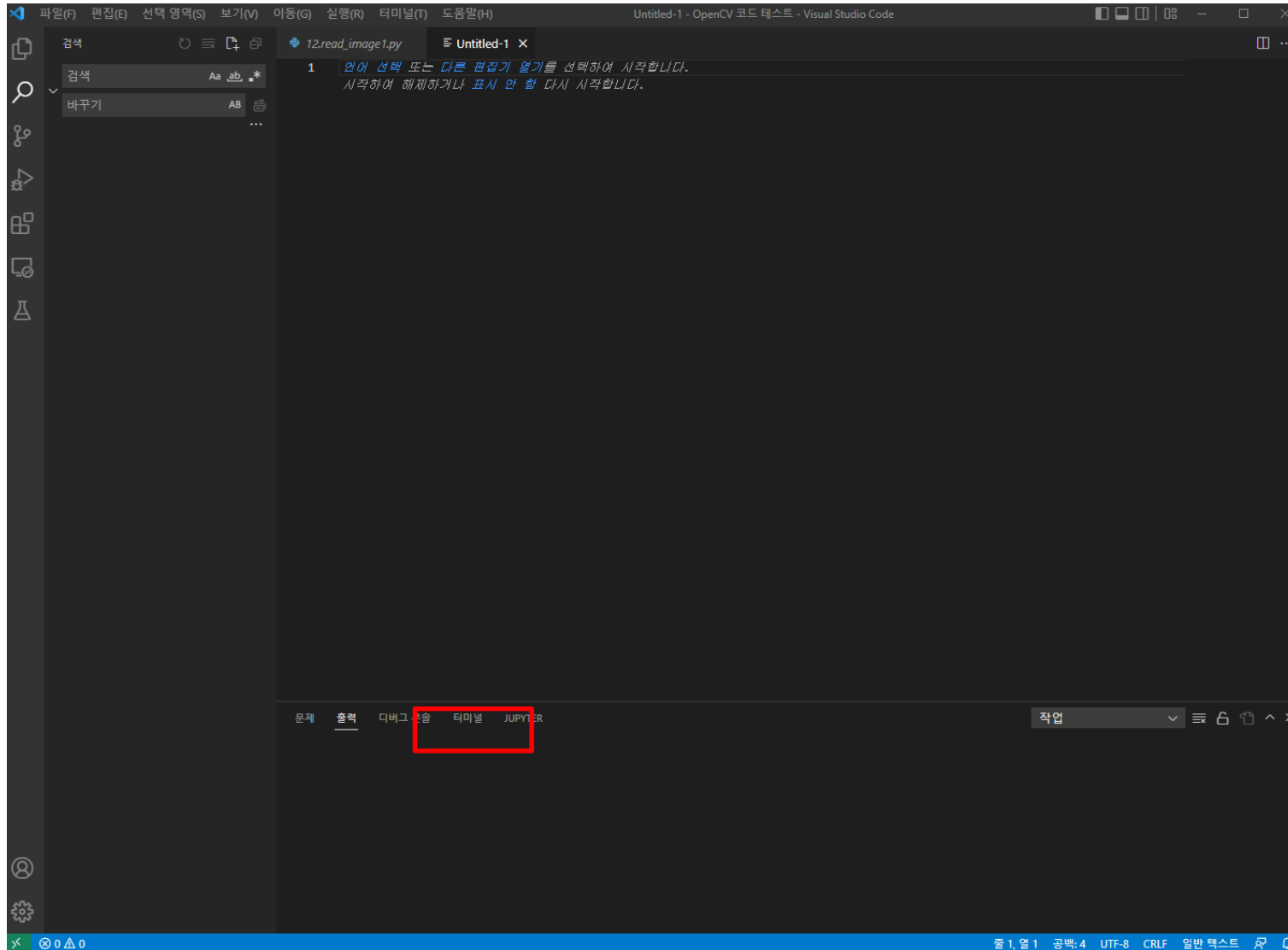
■ 파이썬 설치

- Python 검색 후 설치



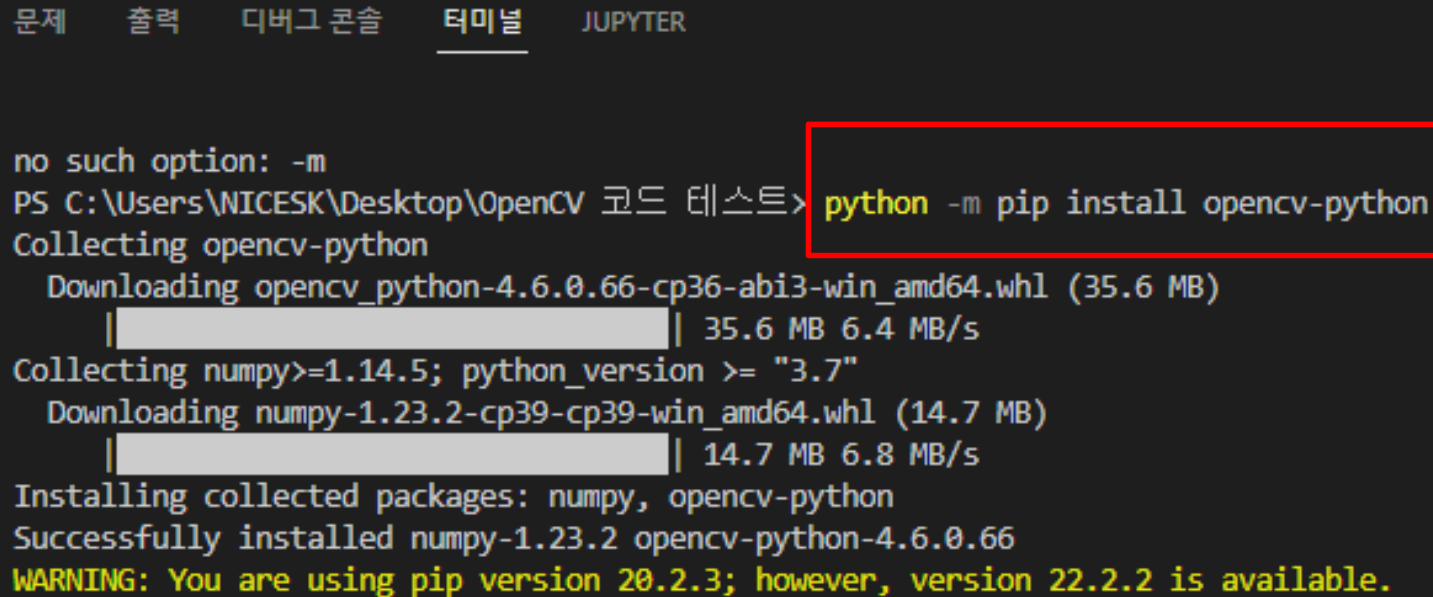
Visual Code 설치

Visual Code → 터미널 선택



Visual Code 설치

1. 설치할 폴더 선택 후,
2. `python -m pip install opencv-python` 입력



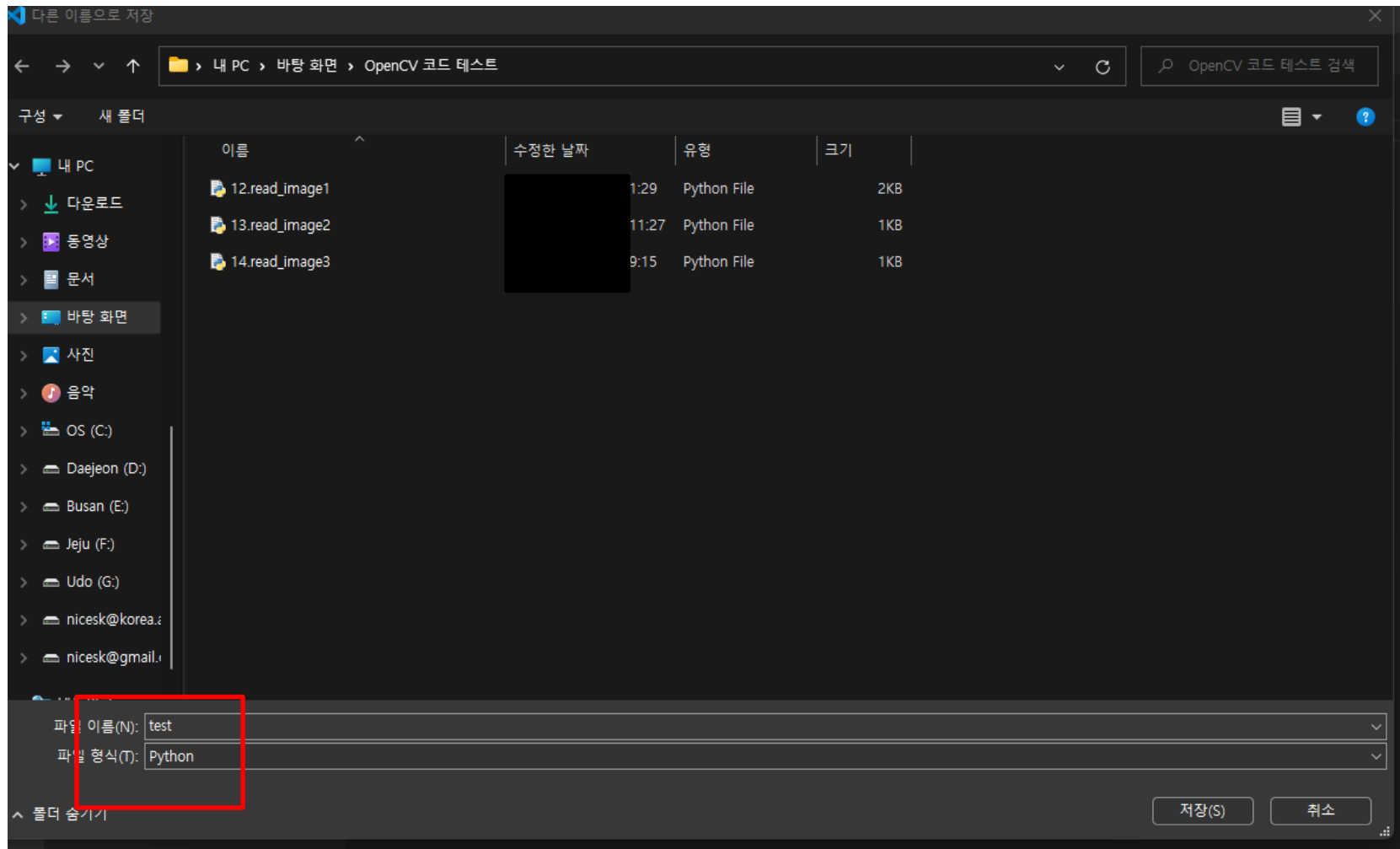
```
문제  출력  디버그 콘솔  터미널  JUPYTER

no such option: -m
PS C:\Users\NICESK\Desktop\OpenCV 코드 테스트> python -m pip install opencv-python
Collecting opencv-python
  Downloading opencv_python-4.6.0.66-cp36-abi3-win_amd64.whl (35.6 MB)
    |████████████████████| 35.6 MB 6.4 MB/s
Collecting numpy>=1.14.5; python_version >= "3.7"
  Downloading numpy-1.23.2-cp39-cp39-win_amd64.whl (14.7 MB)
    |████████████████████| 14.7 MB 6.8 MB/s
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.23.2 opencv-python-4.6.0.66
WARNING: You are using pip version 20.2.3; however, version 22.2.2 is available.
```


Visual Code 설치

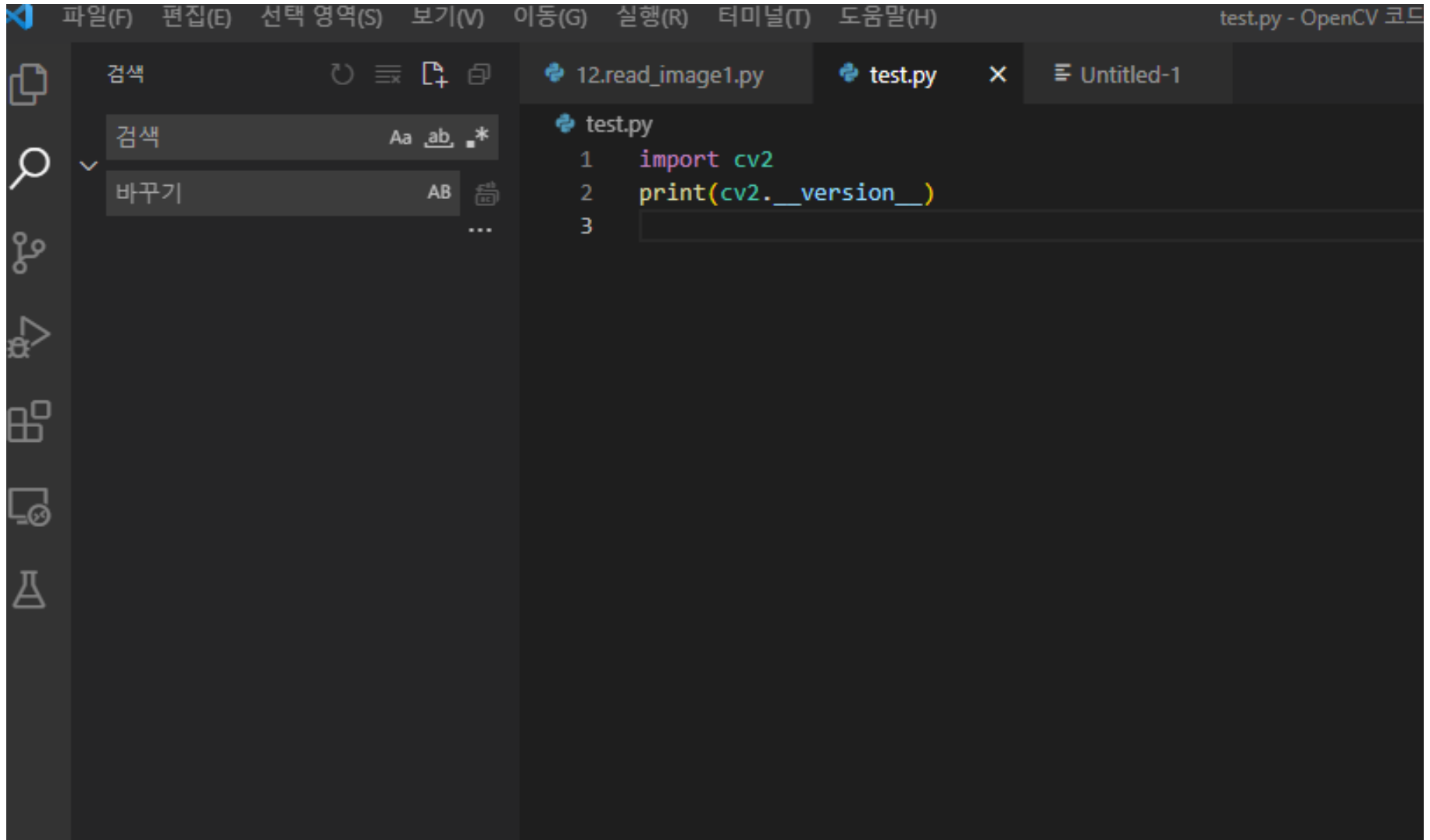
👤 OpenCV 설치 테스트

👤 파일 → 새 텍스트 파일 → 'test.py' 저장



OpenCV-Python 및 라이브러리 설치

다음 2줄 작성해 볼 것



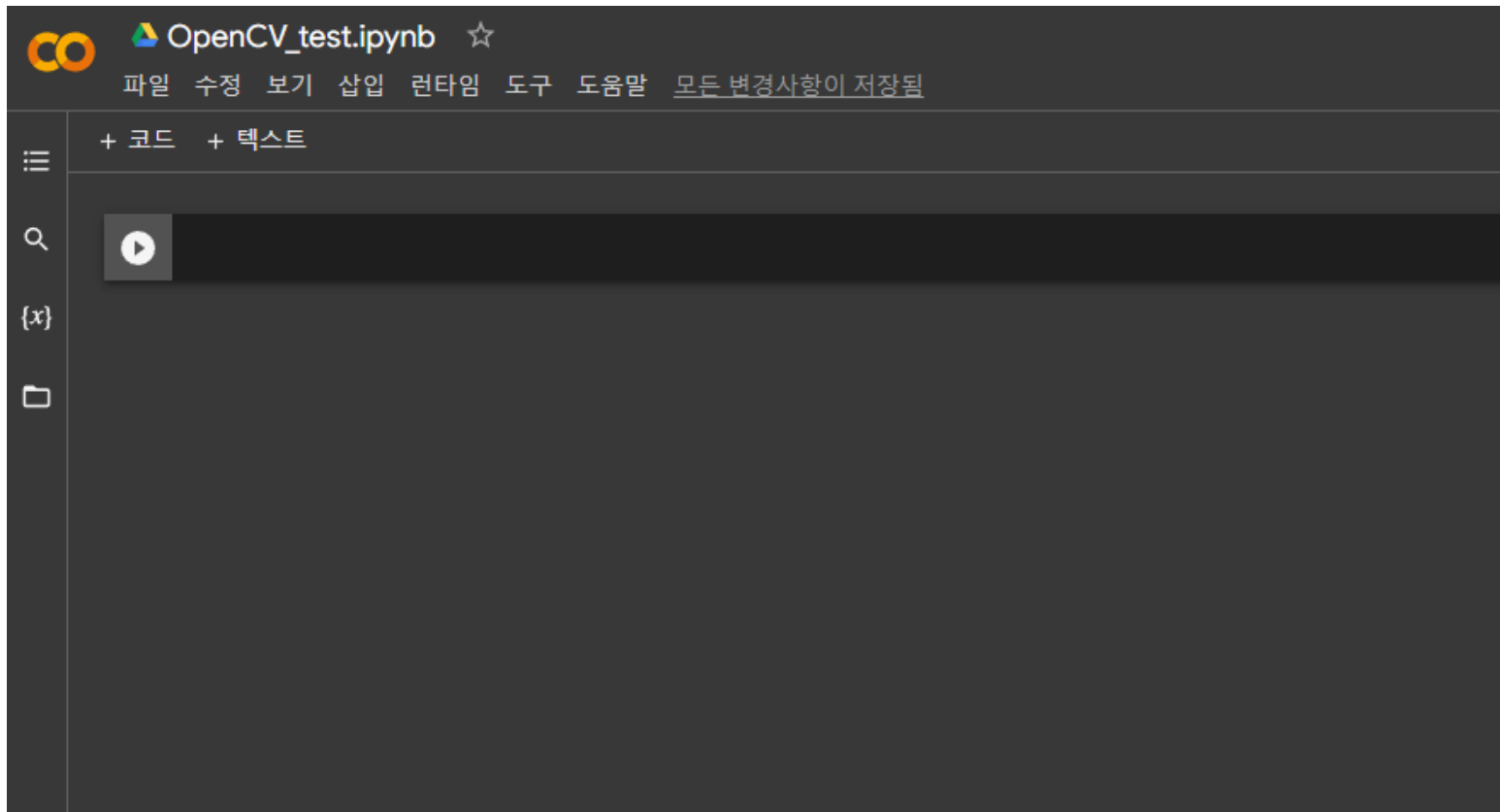
OpenCV-Python 및 라이브러리 설치

출력

```
PS C:\Users\NICESK\Desktop\OpenCV 코드 테스트> & C:/Users/NICESK/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/NICESK/Desktop/OpenCV 코드 테스트/test.py"
4.6.0
PS C:\Users\NICESK\Desktop\OpenCV 코드 테스트>
```

Colab을 이용한 OpenCV-Python 1

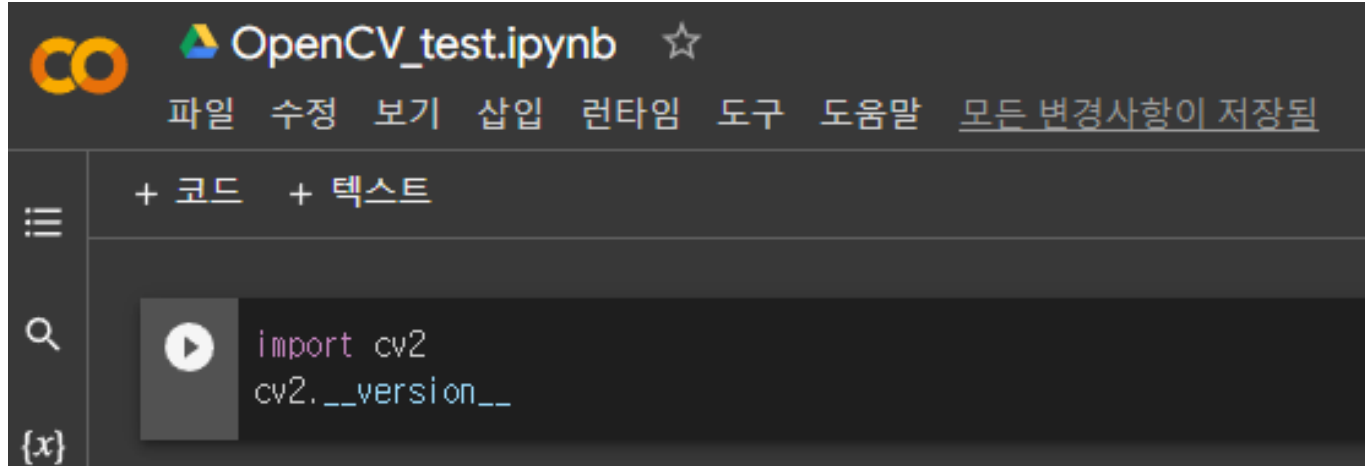
■ 구글 홈페이지에서 Colab을 연결함



■ 파일명: OpenCV_test.ipynb (원하는 저장명을 설정하시기 바람)

Colab을 이용한 OpenCV-Python 2

■ OpenCV 버전 확인하기

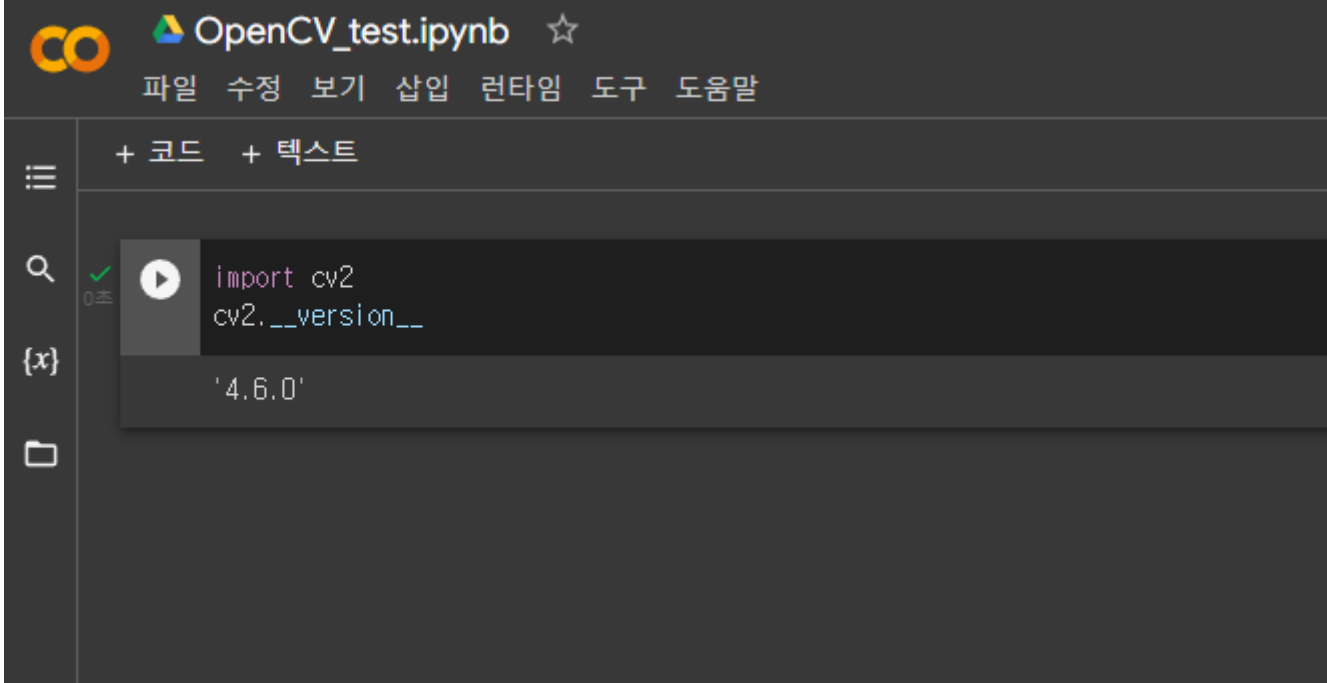


The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled "OpenCV_test.ipynb" with a star icon. Below the title is a menu bar with options: "파일" (File), "수정" (Edit), "보기" (View), "삽입" (Insert), "런타임" (Runtime), "도구" (Tools), "도움말" (Help), and "모든 변경사항이 저장됨" (All changes are saved). On the left side, there is a sidebar with icons for a menu, search, and a variable "{x}". The main area displays a code cell with a play button icon and the following Python code:

```
import cv2
cv2.__version__
```

Colab을 이용한 OpenCV-Python 3

■ OpenCV 버전 확인하기



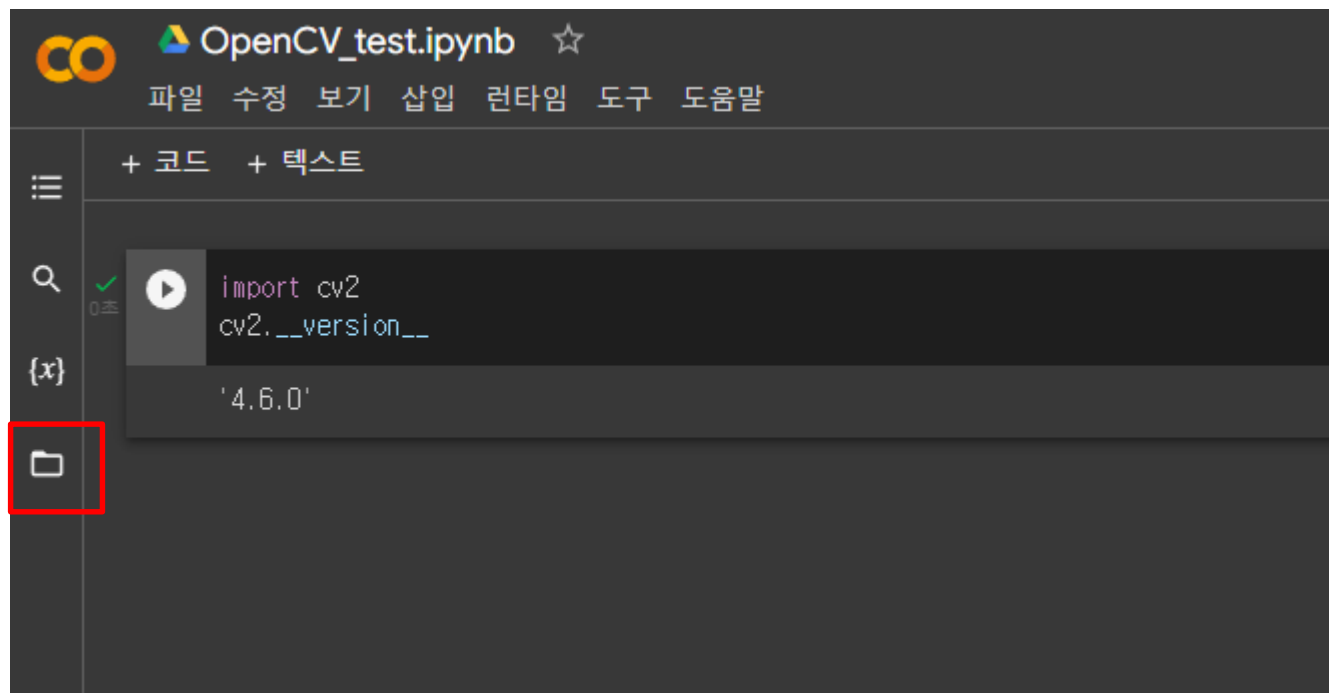
The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled "OpenCV_test.ipynb" with a star icon. Below the title is a menu bar with options: "파일" (File), "수정" (Edit), "보기" (View), "삽입" (Insert), "런타임" (Runtime), "도구" (Tools), and "도움말" (Help). The left sidebar contains icons for a menu, search, variables, and files. The main area shows a code cell with a play button icon, a green checkmark, and the text "0 초". The code in the cell is:

```
import cv2
cv2.__version__
```

The output of the code is displayed below the cell as the string `'4.6.0'`.

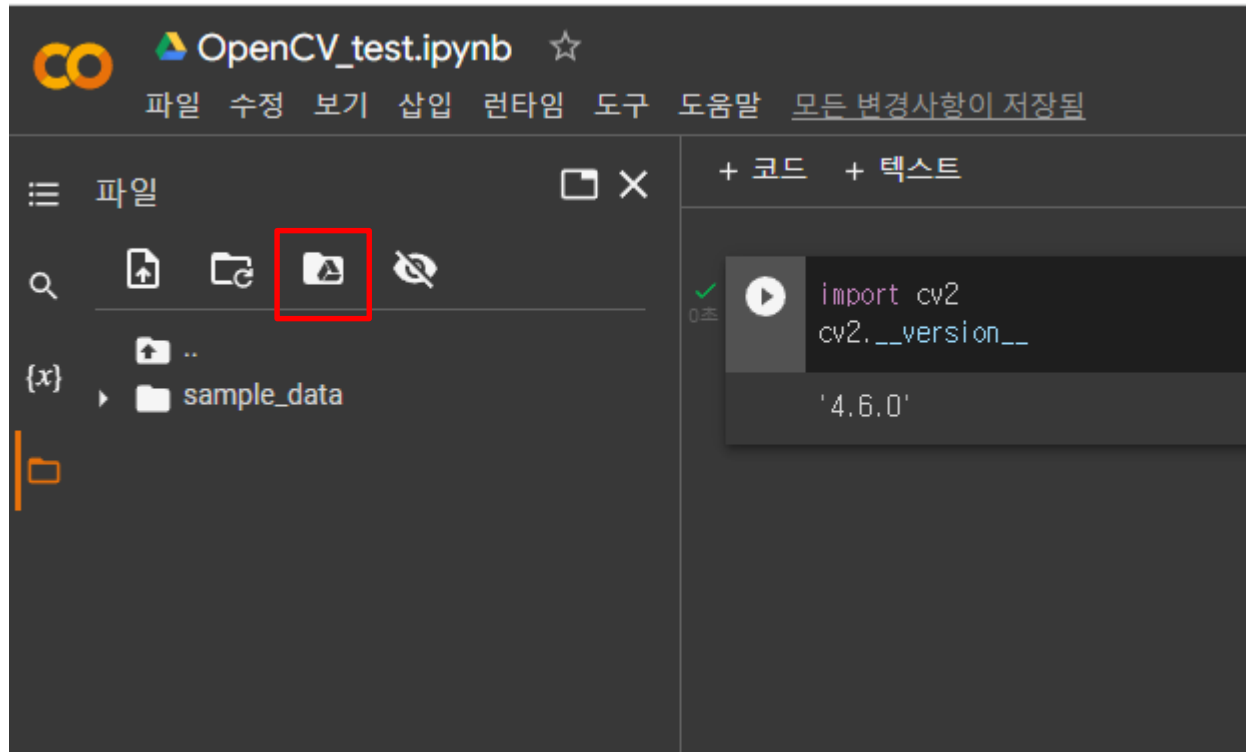
구글 드라이브 마운트 1

■ OpenCV에서 테스트를 위한 이미지 파일 로딩을 위한 구글 드라이브 마운트



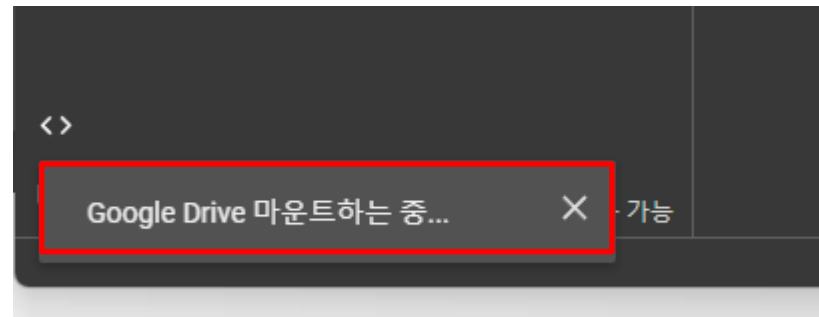
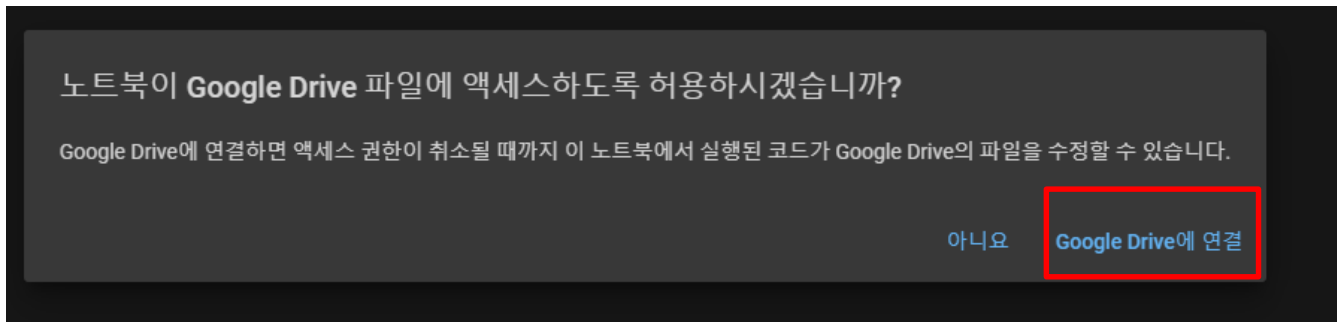
구글 드라이브 마운트 2

■ OpenCV에서 테스트를 위한 이미지 파일 로딩을 위한 구글 드라이브 마운트



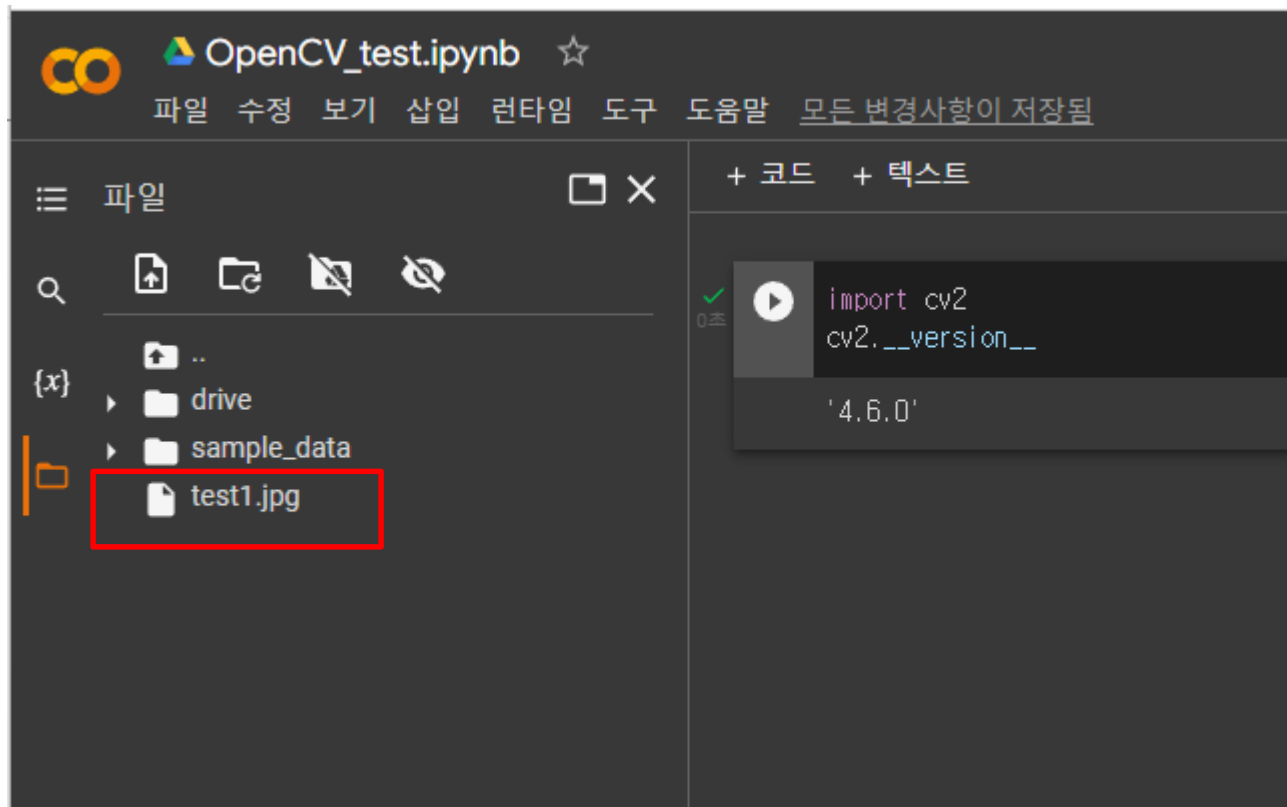
구글 드라이브 마운트 3

■ OpenCV에서 테스트를 위한 이미지 파일 로딩을 위한 구글 드라이브 마운트



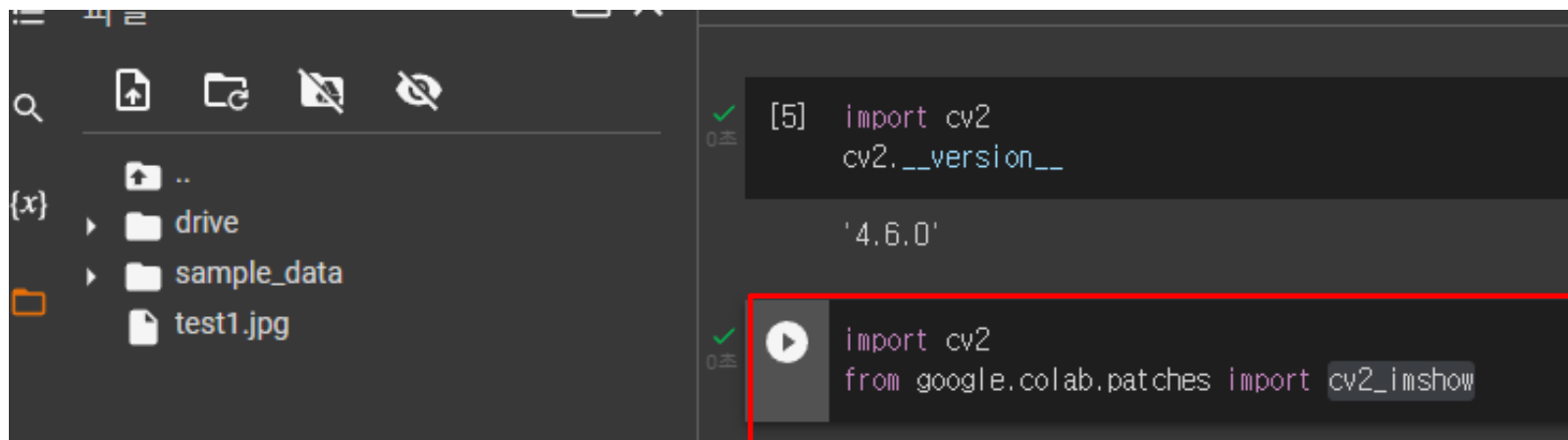
구글 드라이브 마운트 4

■ 이미지를 업로드 (드래그앤드롭) 함

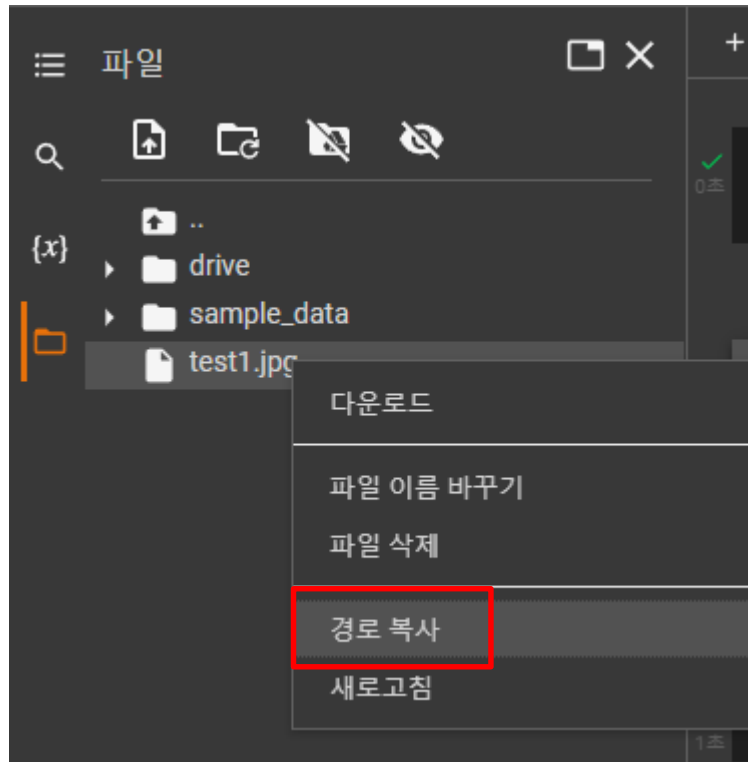


구글 드라이브 마운트 5

■ 이미지의 시각화: cv2_imshow를 import



■ 이미지의 경로 복사



■ 이미지의 경로 복사

A screenshot of a Google Colab notebook interface. It shows two code cells. The first cell contains two lines of Python code: 'import cv2' and 'from google.colab.patches import cv2_imshow'. The second cell contains '[8] path= "/content/test1.jpg"', where the path string is highlighted with a red rectangular box. Both cells show a green checkmark and '0초' (0 seconds) indicating successful execution.

```
import cv2
from google.colab.patches import cv2_imshow

[8] path= '/content/test1.jpg'
```

■ 이미지 로딩

```
[6] import cv2
    from google.colab.patches import cv2_imshow

[8] path='/content/test1.jpg'

image = cv2.imread(path,cv2.IMREAD_COLOR)
```

■ 이미지 시각화: `cv2_imshow(image)`

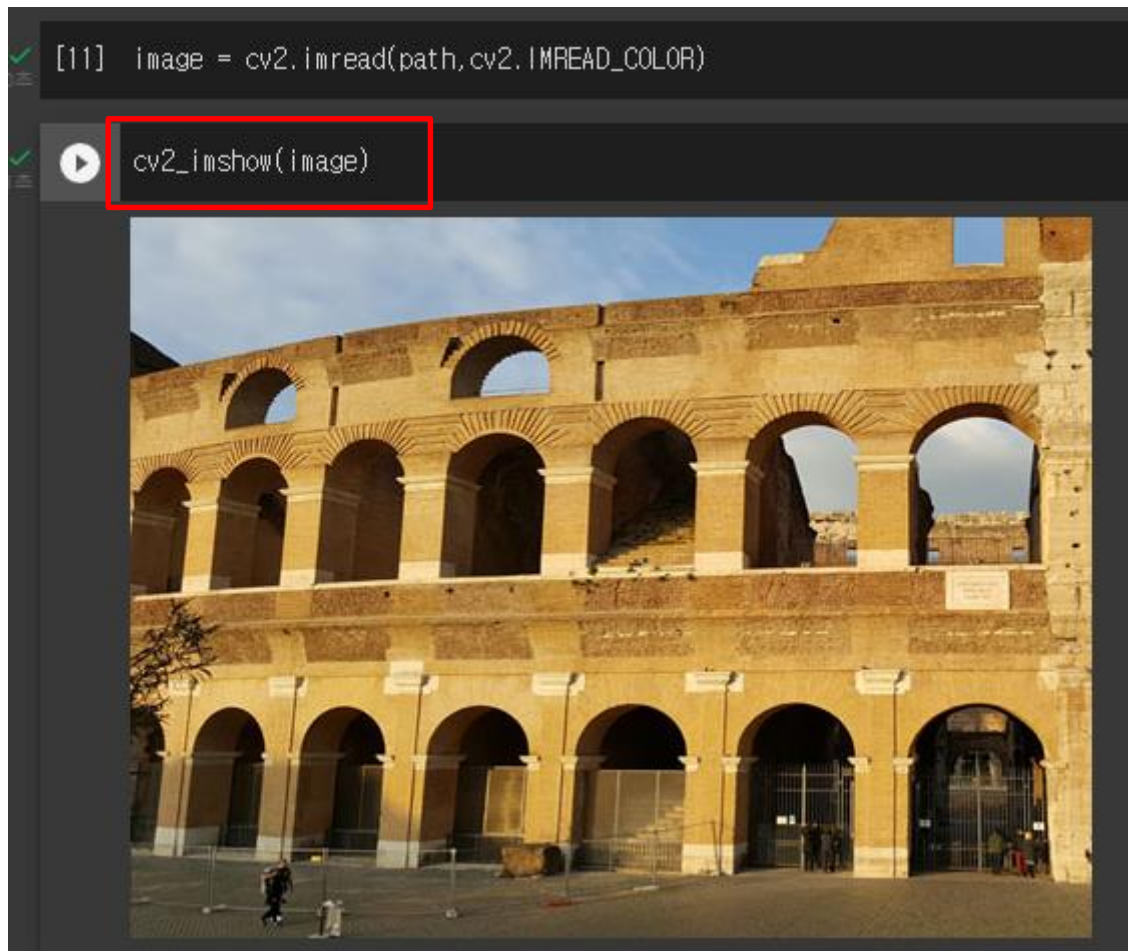
```
def cv2_imshow(a)
```

[탭에서 열기](#) [소스 보기](#)

A replacement for `cv2.imshow()` for use in Jupyter notebooks.

Args:

`a` : `np.ndarray`. shape (N, M) or (N, M, 1) is an NxM grayscale image. shape (N, M, 3) is an NxM BGR color image. shape (N, M, 4) is an NxM BGRA color image.



■ 이미지 시각화: cv2_imshow(image)

```
▶ image = cv2.imread(path, cv2.IMREAD_GRAYSCALE)

if image is not None:
    cv2_imshow(image)
else:
    print("No image file.")
```



이미지 출력

■ 이미지 영상 밝기 변화



```
dst1 = cv2.add(image, 100)  
dst2 = cv2.subtract(image, 100)
```

```
cv2.imshow(image)  
cv2.imshow(dst1)  
cv2.imshow(dst2)
```

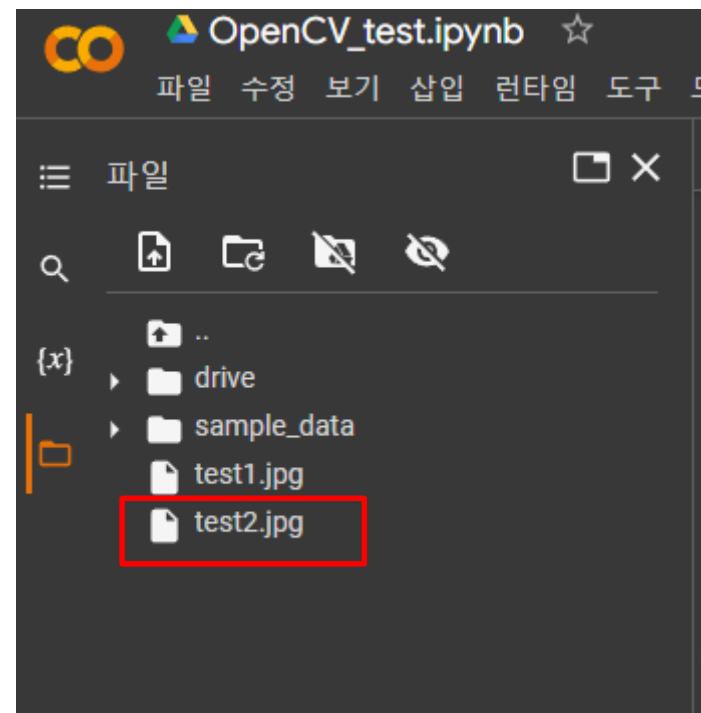


이미지 저장

■ test2.jpg로 이미지 저장

```
[32] cv2.imwrite('/content/test2.jpg', dst1)
```

True



■ test2.jpg 이미지를 보여 주도록 수정해 볼 것

```
[32] cv2.imwrite('/content/test2.jpg', dst1)
```

```
True
```

■ test2.jpg 이미지를 보여 주도록 수정해 볼 것

```
[37] path= '/content/test2.jpg'
```

```
[39] image2 = cv2.imread(path, cv2.IMREAD_COLOR)
```

```
cv2.imshow('image2')
```

