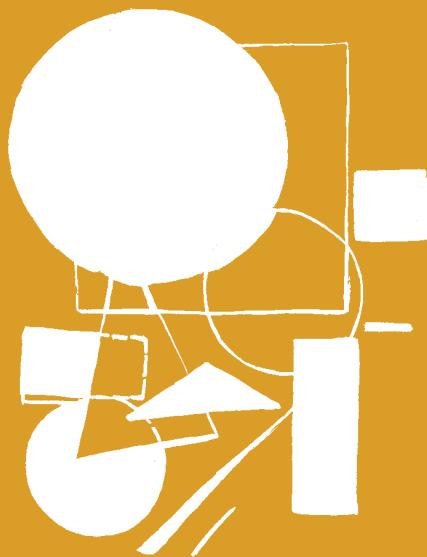


파이썬으로 만드는 OpenCV 프로젝트

프로그래밍 인사이트



파이썬으로 만드는 OpenCV 프로젝트

간단한 영상 입출력부터 머신러닝까지

이세우
지음

인사이트
insight

차례

지은이의 글	xi
베타리더의 글	xiv
1장 개요와 설치	1
1.1 영상 처리와 컴퓨터 비전	1
1.1.1 영상 처리	1
1.1.2 컴퓨터 비전	2
1.2 OpenCV	3
1.2.1 OpenCV 개요	3
1.2.2 OpenCV의 역사	4
1.2.3 필수 개발환경	5
1.3 NumPy 설치	6
1.3.1 윈도우, 맥OS, 우분투	6
1.3.2 라즈베리파이	7
1.4 PC에서 OpenCV-Python 설치	8
1.4.1 공식 배포 바이너리 설치(윈도우)	8
1.4.2 pip로 설치(윈도우, 맥OS, 우분투)	10
1.5 라즈베리파이에서 OpenCV-Python 설치	12
1.5.1 APT로 OpenCV-Python 설치	12
1.5.2 소스 코드 빌드 설치	13
1.5.3 패키지 설치 파일로 설치	19
1.6 OpenCV 공식 문서	20
2장 기본 입출력	23
2.1 이미지와 비디오 입출력	23
2.1.1 이미지 읽기	23
2.1.2 이미지 저장하기	26
2.1.3 동영상 및 카메라 프레임 읽기	27
2.1.4 동영상 파일 읽기	29
2.1.5 카메라(웹캠) 프레임 읽기	30
2.1.6 카메라 비디오 속성 제어	32

2.1.7 비디오 파일 저장하기	34
2.2 그림 그리기	38
2.2.1 직선 그리기	38
2.2.2 사각형 그리기	40
2.2.3 다각형 그리기	42
2.2.4 원, 타원, 호 그리기	44
2.2.5 글씨 그리기	46
2.3 창 관리	49
2.4 이벤트 처리	51
2.4.1 키보드 이벤트	51
2.4.2 마우스 이벤트	52
2.4.3 트랙바	57
 3장 NumPy와 Matplotlib	61
3.1 NumPy	61
3.1.1 이미지와 NumPy	61
3.1.2 NumPy 배열 생성	64
3.1.3 값으로 생성	64
3.1.4 크기와 초기 값으로 생성	66
3.1.5 시퀀스와 난수로 생성	70
3.1.6 dtype 변경	72
3.1.7 차원 변경	73
3.1.8 브로드캐스팅 연산	75
3.1.9 인덱싱과 슬라이싱	78
3.1.10 펜시 인덱싱	81
3.1.11 병합과 분리	82
3.1.12 검색	87
3.1.13 기초 통계 함수	91
3.1.14 이미지 생성	93
3.2 Matplotlib	94
3.2.1 Matplotlib 설치	95
3.2.2 plot	96
3.2.3 color와 style	97
3.2.4 subplot	100
3.2.5 이미지 표시	101

4장 이미지 프로세싱 기초 105

4.1 관심영역 105
4.1.1 관심영역 지정 105
4.1.2 마우스로 관심영역 지정 108
4.2 컬러 스페이스 112
4.2.1 디지털 영상의 종류 112
4.2.2 RGB, BGR, RGBA 114
4.2.3 컬러 스페이스 변환 117
4.2.4 HSV, HSI, HSL 120
4.2.5 YUV, YCbCr 122
4.3 스레시홀딩 124
4.3.1 전역 스레시홀딩 124
4.3.2 오츠의 알고리즘 127
4.3.3 적응형 스레시홀드 130
4.4 이미지 연산 132
4.4.1 영상과 영상의 연산 132
4.4.2 알파 블렌딩 135
4.4.3 비트와이즈 연산 139
4.4.4 차영상 142
4.4.5 이미지 합성과 마스킹 144
4.5 히스토그램 152
4.5.1 히스토그램 계산과 표시 152
4.5.2 노멀라이즈 155
4.5.3 이퀄라이즈 158
4.5.4 CLAHE 161
4.5.5 2D 히스토그램 163
4.5.6 역투영 165
4.5.7 히스토그램 비교 168
4.6 실전 워크숍 170
4.6.1 반해골 괴물 얼굴 합성 170
4.6.2 모션 감지 CCTV 172

5장 기하학적 변환 175

5.1 이동, 확대/축소, 회전 175
5.1.1 이동 176
5.1.2 확대/축소 179

5.1.3 회전	182
5.2 뒤틀기	186
5.2.1 어핀 변환	186
5.2.2 원근 변환	187
5.2.3 삼각형 어핀 변환	192
5.3 렌즈 왜곡	195
5.3.1 리매핑	196
5.3.2 오목 렌즈와 볼록 렌즈 왜곡	199
5.3.3 방사 왜곡	203
5.4 실전 워크숍	206
5.4.1 모자이크 처리 1	206
5.4.2 포토샵 리퀴파이 도구	208
5.4.3 왜곡 거울 카메라	211
 6장 영상 필터	215
6.1 컨볼루션과 블러링	215
6.1.1 필터와 컨볼루션	215
6.1.2 평균 블러링	217
6.1.3 가우시안 블러링	219
6.1.4 미디언 블러링	222
6.1.5 바이레터럴 필터	223
6.2 경계 검출	225
6.2.1 기본 미분 필터	225
6.2.2 로버츠 교차 필터	228
6.2.3 프리윗 필터	229
6.2.4 소벨 필터	230
6.2.5 샤르 필터	232
6.2.6 라플라시안 필터	233
6.2.7 캐니 엣지	234
6.3 모폴로지	236
6.3.1 침식 연산	237
6.3.2 팽창 연산	239
6.3.3 열림과 닫힘, 그밖의 모폴로지 연산	240
6.4 이미지 피라미드	244
6.4.1 가우시안 피라미드	244
6.4.2 라플라시안 피라미드	245

6.5 실전 워크숍	247
6.5.1 모자이크 처리 2	247
6.5.2 스캐치 효과 카메라	248
<hr/>	
7장 영상 분할	251
7.1 컨투어	251
7.1.1 이미지 모멘트와 컨투어 속성	257
7.1.2 컨투어 단순화	263
7.1.3 컨투어와 도형 매칭	266
7.2 허프 변환	268
7.2.1 허프 선 변환	268
7.2.2 확률적 허프 선 변환	271
7.2.3 허프 원 변환	272
7.3 연속 영역 분할	274
7.3.1 거리 변환	274
7.3.2 연결 요소 레이블링	276
7.3.3 색 채우기	278
7.3.4 워터세드	281
7.3.5 그랩컷	284
7.3.6 평균 이동 필터	287
7.4 실전 워크숍	290
7.4.1 도형 알아맞히기	290
7.4.2 문서 스캐너	291
7.4.3 동전 개수 세기	294
<hr/>	
8장 영상 매칭과 추적	301
8.1 비슷한 그림 찾기	301
8.1.1 평균 해시 매칭	302
8.1.2 템플릿 매칭	307
8.2 영상의 특징과 키 포인트	310
8.2.1 코너 특징 검출	311
8.2.2 키 포인트와 특징 검출기	315
8.2.3 GFTTDetector	317
8.2.4 FAST	318
8.2.5 SimpleBlobDetector	320

8.3 디스크립터 추출기	323
8.3.1 특징 디스크립터와 추출기	323
8.3.2 SIFT	324
8.3.3 SURF	326
8.3.4 ORB	328
8.4 특징 매칭	329
8.4.1 특징 매칭 인터페이스	330
8.4.2 BFMatcher	332
8.4.3 FLANN	336
8.4.4 좋은 매칭점 찾기	341
8.4.5 매칭 영역 원근 변환	345
8.5 객체 추적	353
8.5.1 동영상 배경 제거	353
8.5.2 옵티컬 플로	357
8.5.3 MeanShift 추적	362
8.5.4 CamShift 추적	365
8.5.5 Tracking API	367
8.6 실전 워크숍	370
8.6.1 파노라마 사진 생성기	370
8.6.2 책 표지 검색기	372
 9장 머신러닝	377
9.1 OpenCV와 머신러닝	377
9.1.1 머신러닝	378
9.1.2 OpenCV와 머신러닝	379
9.2 k-means 클러스터	381
9.2.1 k-means 알고리즘	381
9.2.2 숫자 손글씨 군집화	386
9.3 k-NN	390
9.3.1 k-NN 알고리즘	390
9.3.2 손글씨 인식	394
9.4 SVM과 HOG	398
9.4.1 SVM 알고리즘	398
9.4.2 HOG 디스크립터	403
9.4.3 보행자 인식	409

9.5 BOW	412
9.5.1 BOW 알고리즘과 객체 인식	412
9.6 캐스케이드 분류기	420
9.6.1 하르 캐스케이드 얼굴 검출	421
9.6.2 LBPH 얼굴 인식	424
9.7 실전 워크숍	430
9.7.1 얼굴 모자이크	430
9.7.2 한니발 마스크 필터 만들기	432
9.7.3 얼굴 웨곡 필터 만들기	433
 부록 DLIB와 얼굴 랜드마크	437
A.1 DLIB	437
A.1.1 DLIB 설치	437
A.1.2 얼굴 랜드마크 검출	439
A.2 얼굴 랜드마크 응용	441
A.2.1 들판에 삼각분할과 얼굴 스와핑	442
 찾아보기	447

지은이의 글

요즘 초등학생인 제 딸이 가장 재미 있어 하는 것은 ‘스노우’라는 스마트폰 카메라 앱으로 사진을 찍는 것입니다. 화면에 비친 자기 얼굴에 예쁘게 화장을 하기도 하고 우스꽝스런 효과를 주기도 하면서 스마트폰을 내려 놓을 줄을 모릅니다.

영상 처리와 컴퓨터 비전이라는 분야는 프로그래밍 분야 전공 여부와는 상관없이 대부분의 사람들이 어려워하는 분야입니다. 복잡한 수학식으로 쓰여진 알고리즘과 C/C++ 언어를 자유롭게 다룰 수 있어야 하기 때문입니다. 그래서 프로그래밍 분야에서도 웹이나 앱 분야와는 달리 그다지 관심을 두는 사람도 별로 없는 것 같습니다. 이 분야가 어렵지만 않다면 재미로라도 친구나 가족에게 ‘스노우’ 같은 프로그램 하나쯤은 만들어 줄만도 할 텐데 말입니다. 물론 돈도 벌 수 있으면 더할 나위 없겠죠.

저는 웹과 앱을 개발하던 평범한 프로그래머입니다. 학교에서 오로지 학점을 따기 위해 접했던 영상 처리는 저에게도 무척 어려웠고 흥미도 없었습니다. 그런데 요즘 저는 저만의 로봇을 만드는 데 많은 시간을 보내고 있습니다. 로봇을 만들려면 필요한 기술이 무척 많지만 그중 빼놓을 수 없는 기술이 바로 컴퓨터 비전이며, OpenCV는 아주 중요한 라이브러리입니다. 그래서 저는 국내외에 출간된 여러 책과 강좌로 오랜 시간 틈틈이 공부를 했는데 여전히 힘들고 어려웠습니다. 책과 강좌들은 대부분 복잡한 수학식으로 풀어낸 알고리즘 위주이거나 OpenCV의 API를 건조하게 다루는 함수 사용 설명서인 경우가 많았습니다. 결국 제가 필요한 코드를 작성할 수 있게 되는 데까지는 무척 오랜 시간과 노력이 필요했습니다. 저는 이 과정에서 얻은 지식과 노하우를 이해하기 쉽게 정리해서 책으로 엮기로 했습니다.

이 책을 파이썬 언어로 쓴 이유는 독자가 주 언어로 어떤 것을 쓰든 파이썬 언어로 작성된 코드를 이해하고 응용하는 것은 어렵지 않을 것이라고 생각했기 때문입니다. 또 파이썬 언어로 작성한 실습 예제는 메모리 관리 같은 부수적인 코드 없이 구현하려는 로직만 표현하므로 코드가 간결하고 OpenCV를 어떻게 사용하는지에만 집중할 수 있습니다. 흔히 많은 강좌나 책에서 독자에게 요구하는 고등학교 수준의 수학 지식을 이 책은 요구하지 않고 오히려 중학교 수학 수준에서 크게 벗어나지 않는 선에서 설명하려고 노력했습니다. 사실 저는 고등학교 수준의 수학도 무척 어렵다고 생각하기 때문입니다.

공학과 기술을 다루는 책의 문장은 “~이다” 또는 “~하다” 같은 문체로 글을 쓰는 것 이 정석인데, 몇몇 관계자의 만류에도 불구하고 이 책은 “~입니다” 또는 “~합니다”같 은 문체로 썼습니다. 학생이든 직장인이든 지치고 피곤한 가운데 시간을 쪼개어 이 책을 보는 독자들도 많을 텐데, 딱딱하게 사실만 늘어 놓는 말투보다는 누군가 친절 하게 설명해 주는 듯한 말투가 독자의 에너지를 아끼는 데 조금이라도 도움이 될 거 라고 생각했기 때문입니다.

이 책은 모두 9개의 장과 1개의 부록으로 구성되어 있습니다. 1~3장까지는 본론 으로 들어가기 위한 준비 작업으로 OpenCV의 설치와 기본적인 영상 입출력 그리고 NumPy 사용법을 다룹니다. 필요한 부분만 골라서 읽어도 되는 부분입니다. 4장부 터 본격적인 영상 처리와 컴퓨터 비전을 다루는데, 각 장의 끝부분에 그 장에서 다룬 내용을 기반으로 몇 가지 쓸모 있는 주제를 모아서 독자가 스스로 만들어 볼 수 있는 워크숍 코너를 준비했습니다. 책으로 혼자 공부하다 보면 함수 하나하나의 사용법과 원리는 알겠는데 막상 프로그램을 작성하려면 전체적인 그림이 그려지지 않아서 난감한 경우가 많습니다. 워크숍 코너는 바로 그런 답답함을 해결하는 데 도움이 될 겁 니다.

4장의 워크숍은 사람 얼굴과 해골 사진을 합성해서 얼굴의 반은 사람이고 반은 해 골인 영상을 만드는 ‘반해골 괴물 얼굴 합성’과 움직임을 감지하는 ‘모션 감지 CCTV’ 를 준비했습니다. 영상 합성과 차 영상의 응용 사례를 직접 구현하면서 영상을 더하고 빼는 기초적인 이미지 연산을 습득할 수 있습니다. 5장의 워크숍은 마우스로 선택한 영역을 모자이크 효과를 내는 ‘모자이크 처리 1’, 얼굴이나 물체를 액체 괴물처럼 편집하는 ‘포토샵 리퀴파이 도구’, 훌쭉이와 뚱뚱이로 만들어 주는 ‘왜곡 거울 카 메라’ 등으로 재미있고 흥미로운 기하학적 변환의 세계로 초대합니다. 6장에서는 컨 볼루션 연산을 이용한 블러링 필터를 사용하는 ‘모자이크 처리 2’, 경계 검출을 활용 해서 ‘스케치 효과 카메라’를 만들어 보면서 컨볼루션 연산 필터와 엣지 검출이라는 매우 중요한 내용을 정리합니다. 7장의 ‘동전 개수 세기’ 워크숍에서는 컨투어와 다 양한 영상 분할 방법들이 어떻게 종합적으로 어우러지는지 이해하는 데 큰 도움이 될 겁니다. 8장의 ‘파노라마 사진 생성기’와 ‘책 표지 검색기’ 워크숍에서는 특징점과 매칭 기술이 객체 인식에 어떻게 활용되는지 정리할 수 있습니다. 9장에는 얼굴을 찾아서 자동으로 모자이크 해주는 ‘얼굴 모자이크’, 얼굴에 한니발 마스크를 씌워주 는 ‘한니발 마스크 필터 만들기’, 얼굴의 눈, 코, 입을 모아주고 눈만 크게 만드는 등 의 ‘얼굴 왜곡 필터 만들기’ 워크숍이 준비되어 있습니다. 부록에서는 DLIB로 얼굴

랜드 마크를 구해서 두 사람의 얼굴을 뒤바꾸는 ‘얼굴 스와핑’이 흥미로울 겁니다. 각장을 마칠 때마다 게임의 퀘스트를 깨는 듯한 재미도 있을 테니 풀이를 보지 말고 꼭 스스로 작성해 보길 권합니다.

최근 머신러닝과 딥러닝의 열기가 뜨겁습니다. 현재를 포함해서 미래에는 프로그래밍의 패러다임을 머신러닝이 주도할 것이라는 것은 분명합니다. 특히 컴퓨터 비전 분야는 딥러닝의 활용 사례로 가장 자주 등장하는 분야입니다. 딥러닝으로 문제를 해결하기 위해서는 데이터의 특성을 잘 이해해야 하는데, 이 책으로 영상 데이터의 특성을 이해하고 컴퓨터 비전 문제를 딥러닝 기법으로 해결하는 데 필요한 실마리를 얻을 수 있을 겁니다.

스마트폰이 보급된 이후로 사진과 동영상만큼 흔한 것이 없습니다. 그것은 프로그래밍 분야에서 영상을 재료로 다뤄야 하는 경우가 점점 더 많아질 것이라는 뜻이기도 합니다. 부디 이 책이 우리나라 영상 관련 프로그래머들의 애근과 삽질을 줄이는 데 조금이나마 보탬이 되길 바랍니다.

끌으로, 이 책이 세상에 나오는 데 노력해 주신 모든 관계자분들과 제가 참고했던 모든 글과 소스 코드의 저작자에게 감사의 마음을 전합니다.

2019년 봄
이세우 쓴

베타리더의 글

책을 전부 읽어보았는데 내용과 구성이 참 좋습니다. 다만 리뷰를 하면서 들었던 생각은 각 OpenCV 함수들 및 파라미터에서 값에 변화를 주면 결과가 어떻게 바뀌는지를 보여주었더라면 독자들이 더 깊이 있게 책을 읽을 수 있었을 것이라는 생각이 듭니다. 하지만 그랬다면 책이 지금보다 몇 배는 더 두꺼워졌을 것이라는 생각은 합니다.

현재의 책은 OpenCV의 가장 기초적인 내용부터 머신러닝까지 폭넓게 다루고 있어서 OpenCV를 처음하시는 분들도 쉽게 고수가 될 수 있습니다. 게다가 영상 처리와 컴퓨터 비전에서 다루는 어려운 개념들을 직관적으로 쉽게 설명하고 있어서 비전공자도 금방 이해할 수 있을 것입니다. 특히, OpenCV를 파이썬으로 다루고 있어 일거양득의 학습 효과를 누릴 수 있습니다. 향후 이 책이 컴퓨터 비전 분야에서 바이블이 되길 바랍니다.

신동혁(EVAR의 CSO)

평소 절친한 형이자 옛 동료인 저자에게서 OpenCV에 관한 본 책의 베타리딩을 부탁 받았을 때 수락하는 것이 망설여졌습니다. 학부생 시절에 보았던 컴퓨터 비전의 복잡한 수학 공식들과 C/C++로 구현한 행렬 계산의 추억들이 떠오른 이유도 있고, 공들여 집필한 내용을 짧은 기간 동안 빠르게 이해하여 베타리더의 역할을 다할 수 있을까 하는 의문이 들었기 때문이기도 합니다. 하지만 책을 읽어나가면서 그런 의문과 어려움은 많이 해소되어 갔습니다.

오랜 강의 경력이 있는 저자가 마치 책을 통해 강의를 하듯 쉽게 설명하고 있으며, 실용적인 예제를 통해 흥미를 유발할 뿐만 아니라 파이썬을 통해 쉽고 빠르게 구현 할 수 있도록 안내하고 있기 때문입니다.

컴퓨터 비전과 OpenCV에 대한 입문자뿐만 아니라 비전 분야에 대한 막연한 두려움을 갖고 계신 독자들에게 이 책은 쉽고 빠르게 접근할 수 있는 훌륭한 가이드가 될 것으로 생각합니다.

윤태희(LG전자)

영상 처리에 대해서 조만간 공부해야겠다고 생각하고 있었기 때문에 리뷰 부탁은 반갑기도 했지만 부담이 크기도 했습니다. 전혀 문외한인 내가 과연 시간 내에 리뷰를 할 수 있을지 걱정이었기 때문입니다. 하지만, 걱정은 이내 사라졌습니다. 초보인 입장에서 4장을 넘어가기가 쉽지 않지만, 4장을 학습하고 나면 이후부터는 술술 넘어갑니다. 8장까지 영상 처리 자체에 대해 충분히 보여주고, 마지막 9장에 영상 머신러닝을 한 장만 할애한 것은 매우 적절했다고 생각합니다. 기초가 확실해야 다음 단계로 갈 수 있는 법이니까요.

항상 남을 가르치는 것을 좋아했던 저자의 책답게, 독자를 체계적으로 안내하며, 간결하고 군더더기가 없습니다. 꼭 필요한 내용만 담아서 부족하지도 넘치지도 않으며, 이 분야를 시작하는 초보에게 매우 적절한 교재라고 생각합니다.

이제 첫 번째 책을 펴낸 저자에게 응원을 보냅니다. 빨리 다음 책을 내길 바랍니다. 그래야 제가 학습하는 데 시간이 덜 들 테니까요.

조현선(마인드웨어웍스의 CTO)

1장

o p e n c v

개요와 설치

이 장에서는 영상 처리와 컴퓨터 비전 그리고 OpenCV가 무엇인지 알아보고 파이썬으로 OpenCV 라이브러리를 사용하여 개발하려면 어떻게 해야 하는지 알아봅니다.

1.1 영상 처리와 컴퓨터 비전

이 절에서는 영상 처리, 이미지 프로세싱 그리고 컴퓨터 비전과 같은 용어를 정리해 봅니다.

1.1.1 영상 처리

영상 처리는 말 그대로 영상을 처리한다는 말입니다. 여기서 처리는 연산을 뜻합니다. 카메라로 찍은 사진 또는 영상에 여러 가지 연산을 가해서 원하는 결과를 새롭게 얻어내는 과정을 영상 처리 또는 이미지 프로세싱(image processing)이라고 합니다. 대부분 영상 처리의 목적은 더 좋은 품질의 영상을 얻으려는 것입니다. 몇 가지 예를 들면 다음과 같습니다.

- 영상(화질) 개선: 사진이나 동영상이 너무 어둡거나 밝아서 화질을 개선하는 과정
- 영상 복원: 오래되어 빛바랜 옛날 사진이나 영상을 현대적인 품질로 복원하는 과정
- 영상 분할: 사진이나 영상에서 원하는 부분만 오려내는 과정

이와 같은 예에 가장 적합한 도구를 생각하면 바로 떠오르는 것이 어도비(Adobe)사의 포토샵(Photoshop)일 것입니다. OpenCV는 포토샵에 있는 여러 기능을 프로그래밍 언어로 구현할 수 있게 해주는 라이브러리라고 생각해도 크게 틀리지 않습니다.

1.1.2 컴퓨터 비전

컴퓨터 비전은 영상 처리 개념을 포함하는 좀 더 큰 포괄적인 의미입니다. 영상 처리가 원본 영상을 사용자가 원하는 새로운 영상으로 바꿔 주는 기술이라면 컴퓨터 비전은 영상에서 의미 있는 정보를 추출해 주는 기술을 말합니다. 예를 들면 다음과 같습니다.

- 객체 검출(object detection): 영상 속에 원하는 대상이 어디에 있는지 검출
- 객체 추적(object tracking): 영상 속 관심 있는 피사체가 어디로 움직이는지 추적
- 객체 인식(object recognition): 영상 속 피사체가 무엇인지 인식

컴퓨터 비전 작업을 하기 전에 영상 처리 작업을 하는 경우가 많습니다. 만약 영상에서 객체를 인식하려고 하는데, 화질이 나쁘면 당연히 인식이 잘 되지 않을 것입니다. 그래서 먼저 화질 개선 작업을 해야 할 수도 있습니다. 하지만, 컴퓨터 비전의 전처리 작업에 화질 개선 작업만 있는 것은 아닙니다. 오히려 고화질 영상은 물체를 인식하는 데 불필요하게 연산이 많이 필요하므로 영상을 단순화하는 작업이 필요할 수도 있습니다.



[그림 1-1] 객체 인식을 예로 든 컴퓨터 비전 프로세스

일반적으로 컴퓨터 비전 작업은 입력받은 원본 영상을 영상 처리하여 원하는 품질의 결과 영상을 얻어낸 다음, 컴퓨터 비전으로 원하는 정보를 얻어내는 과정이 반복적으로 일어납니다.

또한, 컴퓨터 비전의 대표적인 분야인 객체 인식이나 객체 검출 작업은 잘 정제되고 축적된 학습 자료를 토대로 이루어지는데, 이 과정에서 머신 러닝 분야에서 사용하는 다양한 알고리즘을 사용해야 합니다. 이런 이유로 최근 OpenCV는 머신 러닝 분야와 관련성이 높아지고 있으며, 관련 기능이 빠르게 추가되고 있습니다.

1.2 OpenCV

1.2.1 OpenCV 개요

OpenCV(오픈씨브이)는 오픈 소스 컴퓨터 비전 라이브러리(Open Source Computer Vision Library)를 줄여 쓴 말입니다. OpenCV는 영상 처리와 컴퓨터 비전 프로그래밍 분야의 가장 대표적인 라이브러리입니다.

예전에는 프로그래밍 영역 중에서도 영상 처리나 컴퓨터 비전 분야는 대학원에서나 본격적으로 접하게 될 만큼 비교적 전문적인 분야였습니다. 매우 복잡한 수학적 알고리즘을 C/C++ 언어로 구현해야 했으므로 체계적인 수학 지식과 뛰어난 프로그래밍 실력을 갖추고 있지 않으면 이해하기 어려웠기 때문입니다.

하지만, 이제는 OpenCV와 같은 훌륭한 라이브러리가 있어 기초적인 수학 지식만으로도 이미 구현된 알고리즘을 손쉽게 사용할 수 있게 된 데다가, 메모리 주소 하나 하나를 직접 다뤄야 했던 어렵고 복잡한 C 언어가 아닌 비교적 배우기 쉽고 빠르게 구현할 수 있는 파이썬 언어로 OpenCV를 사용할 수 있게 되면서 영상 처리와 컴퓨터 비전 분야의 문턱이 무척 낮아졌습니다.

OpenCV는 맨 처음에는 C 언어로 작성되었지만, 지금은 C++ 언어를 공식적으로 채택하고 있고, 파이썬, 자바 언어를 바인딩 언어로 공식 지원하고 있습니다. 플랫폼으로는 윈도우, 맥OS, 리눅스는 물론 안드로이드와 iOS까지 지원합니다.

OpenCV의 공식 웹사이트는 다음과 같습니다.

- <http://www.opencv.org>

OpenCV는 소스 코드를 2개의 저장소에 나누어 관리하며, 각각의 저장소 주소는 다음과 같습니다.

- 메인 저장소: <https://github.com/opencv/opencv>
- 엑스트라 저장소: https://github.com/opencv/opencv_contrib

메인 저장소와 엑스트라(extra) 저장소에서 관리하는 각각의 소스 코드는 의미가 조금 다릅니다. 메인 저장소에서는 OpenCV 공식 배포에 사용하는 코드를 관리합니다. 엑스트라 저장소는 컨트리브(contrib) 저장소라고도 하는데, 아직 알고리즘이나 구현의 성숙도가 떨어지거나 대중화되지 않은 내용을 포함하고 있으며, 향후 완성도

가 높아지면 메인 저장소로 옮겨집니다. 또한 엑스트라 저장소에는 특허권을 가지고 있어서 사용에 제약이 있는 알고리즘을 구현한 코드도 포함되어 있는데, 앞으로 다른 SIFT, SURF 등이 여기에 해당합니다.

OpenCV 메인 저장소에서 배포하는 공식 배포판은 BSD 라이선스로 연구와 상업 용도와 무관하게 무료로 사용할 수 있으며, OpenCV를 활용해서 만든 소스 코드를 오픈할 의무가 없습니다. 다만, 엑스트라 저장소의 코드를 포함하는 경우에는 상업 용도의 사용이 제한되니 주의해야 합니다.

1.2.2 OpenCV의 역사

OpenCV는 본래 인텔의 러시아 팀에서 CPU 집약적인 응용프로그램의 성능을 향상시키기 위한 연구의 일부로 시작되었습니다.

OpenCV는 1999년 1월 IPL(Image Process Library)을 기반으로 C 언어로 처음 작성되었으며, 2000년에 첫 알파 버전을 대중에게 선보였습니다. 2001년부터 2005년 까지 다섯 번의 베타 버전을 내놓은 후에 2006년 10월에 첫 번째 정식 버전 1.0을 내놓았는데, 버전 1.0에는 미리 컴파일된 파이썬 모듈이 포함되었고 볼랜드 C++를 위한 CMake 파일도 추가되었습니다.

2009년 9월에 2.0 버전을 배포하면서 STL 기반의 새로운 C++ API와 새로운 파이썬 인터페이스를 포함시켰고, 2010년 12월에 배포한 2.2 버전에서는 패키지를 새롭게 정립하면서 지금의 형태인 core, imgproc, features2d, ml, contrib 등의 패키지 구조를 갖추게 되었습니다. 아울러 파이썬 언어 바인딩에서도 큰 변화가 있었는데, 이전까지는 영상 정보를 저장하는 데 사용하는 자료구조로 자체적인 API를 사용했는데, 이때부터 데이터 과학 분야에 널리 사용하는 NumPy(넘파이) 모듈로 바뀌었습니다.

2011년 7월에 2.3 버전을 배포하면서 안드로이드를 지원하기 시작했으며, 파이썬 바인딩의 모듈 이름이 cv였는데 OpenCV 2.x에 대응해서 cv2로 변경하였습니다. OpenCV는 2.4 버전 이후 패키지의 구조 변경과 폐기된 API 제거 등의 이유로 2015년 6월부터 버전 번호를 3.0으로 바꾸어 출시하였습니다. 이 책을 쓰는 2019년 1월 기준으로 마지막 배포 버전은 4.0입니다.

OpenCV 최신 버전과 이 책의 설치 버전에 대한 안내

이 책을 발간할 무렵 OpenCV는 4.0 버전을 최신 버전으로 공식 배포했습니다. 그런데도 이 책은 최신 버전을 설치하는 내용을 반영하지 않고 3.4.1 버전을 기준으로 집필을 완료했습니다. 그 이유는 크게 두 가지입니다.

첫째, OpenCV 공식 사이트에서는 4.0 버전을 배포하고 있지만, 파이썬 모듈을 설치할 때 사용하는 pip 명령어가 참조하는 Pypi 저장소에는 아직 4.0 버전을 배포하고 있지 않습니다. 따라서 pip 명령어로 OpenCV-Python 4.0 버전을 설치할 수 없고 굳이 설치하려면 독자가 소스 코드를 직접 빌드해야 하는데, 이 책에서 최신 버전에서만 동작하는 기능을 따로 다루지는 않으므로 굳이 그런 수고를 할 필요가 없다고 판단했기 때문입니다.

둘째, Pypi 저장소에서 제공하는 최신 버전은 3.4.4인데 그마저도 사용하지 않고 3.4.1 버전을 사용하는 이유는 OpenCV 3.4.2 버전 이후부터 SIFT와 SURF 등 특허가 있는 알고리즘에 대한 구현 내용을 포함하지 않고 배포하고 있기 때문입니다. 이 알고리즘들을 사용해 보려면 해당 알고리즘을 사용할 수 있도록 하는 옵션으로 OPENCV_ENABLE_NONFREE=ON을 지정해서 독자가 직접 빌드해야 합니다.

이 책의 설치 과정에서 설명하는 opencv-contrib-python 3.4.1.15 버전까지는 SIFT와 SURF 등 특허가 있는 알고리즘도 기본으로 포함되어 있어서 이 책에서 다루는 모든 예제를 별도의 빌드 과정 없이 사용해 볼 수 있습니다. 따라서 별도의 빌드 과정 없이 이 책의 예제를 모두 실행해 보려면 Python3에서 동작하는 아래의 pip3 명령어로 설치할 것을 권장합니다.

```
pip3 install opencv-contrib-python==3.4.1.15
```

1.2.3 필수 개발환경

OpenCV의 파이썬 언어 바인딩은 파이썬 버전 2.7과 3.x에 따른 지원에 차이가 전혀 없으므로 어떤 버전의 파이썬을 사용해도 좋습니다. 다만 OpenCV 파이썬 모듈이 사용하는 NumPy 모듈이 파이썬 3.0부터 3.3까지의 버전을 지원하지 않아 파이썬 3을 사용하려면 파이썬 3.4 버전 이상이 필요합니다.

이제 본격적인 환경 설정과 개발을 위해서 파이썬 버전 2와 3 중에 어느 것을 사용할지 결정해야 합니다. 이 책은 파이썬 3.6과 OpenCV 3.4.1을 기준으로 설명합니다. 필자가 이 책을 쓰면서 사용한 환경을 요약하면 다음과 같습니다. 독자 여러분도 같은 환경을 구성할 것을 권장합니다.

- 파이썬 3.6
- NumPy 모듈 1.14
- OpenCV-Python 모듈 3.4.1, 엑스트라(contrib) 모듈 포함
- Matplotlib 2.2.2

이제 OpenCV를 윈도우, 맥OS, 우분투 운영체제에서 설치하는 방법과 라즈베리파이에 설치하는 방법을 차례대로 설명합니다. 독자가 사용하려는 환경에 맞게 해당 부분을 골라서 읽기 바랍니다. 아래 표는 개발환경을 설치하기 전에 준비되어 있어야 하는 플랫폼별 요구 사항입니다.

플랫폼	버전	필요 하드웨어	필수 소프트웨어
윈도우	윈도우 7 이상	웹캠	파이썬 3.4 이상
맥OS	OS X 시에라 이상	웹캠	파이썬 3.4 이상
우분투	16.04 LTS	웹캠	파이썬 3.4 이상
라즈베리파이	라즈베리파이 3 모델 B 보드 라즈비안 스트레치(Stretch)	SD 카드 16GB 이상 파이 카메라 또는 USB 웹캠(UVC 지원)	

[표 1-1] OpenCV 개발환경 설치 전 필수 요구 사항

1.3 NumPy 설치

OpenCV-Python 모듈은 2 버전부터 NumPy 라이브러리가 없으면 동작하지 않습니다. 그래서 OpenCV를 설치하기 전에 종속 라이브러리인 NumPy를 설치해야 합니다. NumPy 라이브러리는 pip 명령으로 간단히 설치할 수 있습니다.

다음은 Pypi에서 NumPy를 배포하는 웹사이트의 URL입니다.

- <https://pypi.python.org/pypi/numpy>

1.3.1 윈도우, 맥OS, 우분투

윈도우는 pip 명령어로, 맥OS와 우분투는 pip3 명령어로 NumPy를 설치하면 됩니다. 설치 명령어는 아래와 같습니다. 명령어는 pip3로 통일해서 설명하겠습니다.

```
$ pip3 install numpy
```

설치가 완료되면 파이썬 대화형 콘솔을 열어서 NumPy의 버전을 출력해 보면 정상적으로 설치되었는지 확인할 수 있습니다.

```
>>> import numpy
>>> numpy.__version__
'1.14.0'
```

1.3.2 라즈베리파이

라즈베리파이의 경우 PC와 달리 CPU가 ARM 아키텍처이므로 NumPy 공식 바이너리 배포에 포함되어 있지 않습니다. 따라서 라즈베리파이에서는 pip3 명령어로 NumPy를 설치할 수 없으며, 대신 테비안 패키지 관리자인 APT(Advanced Package Tool)를 사용하면 됩니다.

설치 전에 우선 패키지 목록을 업데이트합니다.

```
$ sudo apt-get update
```

설치할 NumPy 패키지 이름은, 파이썬 2.x 버전은 `python-numpy`이고 파이썬 3.x 버전은 `python3-numpy`입니다. 파이썬 3 버전에 맞는 설치 명령어는 아래와 같습니다.

```
$ sudo apt-get install python3-numpy
```

설치가 완료되면 파이썬 대화형 콘솔을 열어 버전 번호를 출력해 보면 정상적으로 설치되었는지 확인할 수 있습니다.

```
>>> import numpy
>>> numpy.__version__
'1.12.1'
```

라즈베리파이의 경우 APT 저장소에서 제공하는 NumPy의 버전이 다소 낮지만, OpenCV를 사용하는 데는 문제가 없습니다. NumPy의 최신 버전을 설치하려면 소스 코드를 다운로드해서 직접 빌드하면 되는데 아래의 URL을 참고하면 됩니다.

- <http://scipy.github.io/devdocs/building/linux.html>

1.4 PC에서 OpenCV-Python 설치

윈도우, 맥OS, 우분투의 경우 OpenCV-Python을 설치하는 방법은 일반적으로 크게 세 가지 정도입니다.

가장 최신의 기능을 가장 손쉽게 사용할 수 있는 방법은 pip 명령어를 이용하는 것 이므로 가급적 pip 명령으로 설치하는 것을 권장합니다.

설치 방법	운영체제	파이썬 버전	OpenCV 버전	OpenCV 패키지	난이도
공식 배포 바이너리	윈도우	2.7, 3.5+	모든 버전	메인 모듈만	쉬움
pip	윈도우, 맥OS, 우분투	2.7, 3.4+	3.x	메인, 엑스트라	쉬움
소스 빌드	윈도우, 맥OS, 우분투	2.7, 3.4+	모든 버전	메인, 엑스트라	어려움

[표 1-2] PC에서 OpenCV-Python 설치 방법

OpenCV에서는 공식 버전을 미리 컴파일된 바이너리 파일로 배포하는데, 윈도우용으로만 제공하고 있습니다.

pip 명령어를 이용하면 OpenCV 3.x 버전을 윈도우, 맥OS, 우분투에 모두 설치할 수 있지만, 대신 OpenCV 2.x 버전은 설치할 수 없습니다. 오래된 OpenCV 2.4 버전을 사용할 것이 아니라면 pip 명령어를 이용하는 것이 가장 탁월한 선택입니다.

소스 코드를 직접 빌드하는 방법은 모든 운영체제와 플랫폼에서 가능하지만, 매우 복잡하고 귀찮은 작업입니다. 이 책에서는 소스 코드를 직접 빌드하는 방법을 설명하지 않습니다. 직접 빌드하는 방법은 아래의 URL을 참고하세요.

- 윈도우: https://docs.opencv.org/3.4.1/d3/d52/tutorial_windows_install.html
- 리눅스: https://docs.opencv.org/3.4.1/d7/d9f/tutorial_linux_install.html

이제 독자가 사용하는 환경에 맞게 골라서 설치를 진행해 봅니다.

1.4.1 공식 배포 바이너리 설치(윈도우)

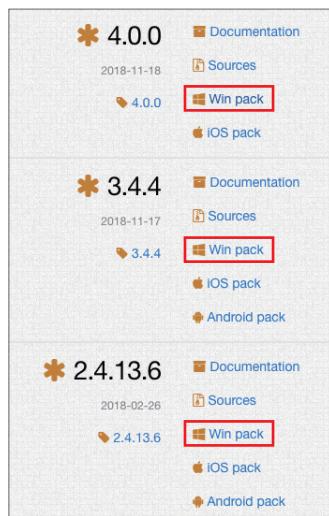
OpenCV 공식 배포 바이너리를 설치하는 것은 OpenCV를 파이썬 언어로 개발하려고 할 때 가장 손쉽게 설치할 수 있는 방법입니다. 그리고 이 방법은 최신 버전인 OpenCV 4.0 버전에서부터 오래된 2.4 버전까지 배포하고 있어서 손쉽게 OpenCV 버전을 바꾸어 가며 작업할 수 있습니다.

하지만, 이 라이브러리는 액스트라 모듈이 포함되지 않은 메인 모듈만을 제공하므로 다양한 최신 알고리즘을 실습해 볼 수 없다는 단점도 있습니다. 또한, 이 방법으로 설치하면 이 책에 수록된 모든 예제를 실습할 수도 없습니다.

OpenCV 공식 배포 바이너리는 아래의 URL에서 무료로 다운로드할 수 있습니다.

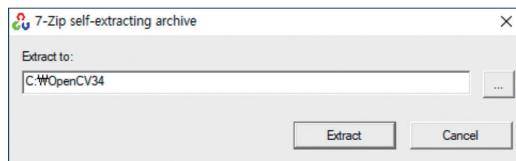
- <https://opencv.org/releases.html>

해당 URL에서 사용하고자 하는 OpenCV 버전을 골라 ‘Win pack’을 다운로드합니다.



[그림 1-2] OpenCV 다운로드 페이지

다운로드한 파일은 더블 클릭하면 자동으로 압축이 풀리는 실행 파일입니다. 압축을 풀 디렉터리를 지정하고 진행하면 해당 디렉터리에 압축이 풀립니다.



[그림 1-3] OpenCV 라이브러리 압축 해제

OpenCV 버전 4.0 이후부터는 압축이 풀리면 다음과 같은 경로에 설치 스크립트가 있습니다.

- opencv\build\bin\python\setup.py

이 파일을 아래의 명령어로 실행합니다.

```
python3 setup.py install
```

OpenCV 3.x 이전 버전인 경우 압축이 풀리면 다음과 같이 2개의 경로에 cv2.pyd 파일이 2개 존재할 것입니다.

- opencv\build\bin\python\2.7\x64\cv2.pyd
- opencv\build\bin\python\2.7\x86\cv2.pyd

각각의 cv2.pyd 파일은 사용하는 파이썬 실행 환경이 32비트인지 혹은 64비트인지에 따라 골라서 사용하도록 제공됩니다. 이 파일 중에 독자가 사용하는 파이썬 환경에 맞는 것을 골라 파이썬이 설치된 디렉터리의 Lib\site-packages 디렉터리에 복사합니다.

만약 파이썬을 c:\Python27에 설치했다면 복사할 최종 경로는 다음과 같습니다.

- c:\Python27\Lib\site-packages\cv2.pyd

이제 파이썬 대화형 콘솔에서 OpenCV-Python의 버전을 확인해 봅니다.

```
>>> import cv2
>>> cv2.__version__
'3.4.1'
```

다시 한번 말씀드리지만, 이 방법은 윈도우 이외의 환경에서는 사용할 수 없으며, 이 책의 실습 예제를 모두 실행할 수도 없습니다.

1.4.2 pip로 설치(윈도우, 맥OS, 우분투)

파이썬의 패키지 관리자인 pip를 이용하면 OpenCV-Python을 손쉽게 설치할 수 있습니다. OpenCV-Python은 메인 모듈만 있는 버전과 엑스트라 모듈을 포함한 버전을 선택해서 설치할 수 있고 파이썬 2.7과 3 버전을 모두 지원합니다. 다만, Pypi에

서 제공하는 OpenCV-Python 모듈은 OpenCV 3.0 이상의 버전만 제공하고 2.4 버전은 지원하지 않습니다. OpenCV-Python 모듈의 Pypi 페이지 URL은 아래와 같습니다.

- 메인 모듈: <https://pypi.python.org/pypi/opencv-python>
- 엑스트라 모듈 포함: <https://pypi.python.org/pypi/opencv-contrib-python>

어느 모듈을 설치할 것인지 사용할 용도에 맞게 선택한 후 pip3 명령어로 설치합니다. 메인 모듈만을 설치할 경우 패키지 이름은 opencv-python입니다.

```
$ pip3 install opencv-python
```

엑스트라 모듈을 포함한 패키지의 이름은 opencv-contrib-python입니다. 이 책에서 설명하는 모든 예제를 실행하려면 opencv-contrib-python을 설치해야 합니다.

```
$ pip3 install opencv-contrib-python
```

pip3 명령어로 opencv-python을 설치하면 가장 최신 버전이 선택되어 설치됩니다. 이때 최신 버전이 아니라 자기가 원하는 특정 버전을 설치하려면 패키지 이름 뒤에 버전 번호를 명시하면 됩니다. 예를 들어 opencv-contrib-python 3.4.1.15 버전을 설치하려면 명령어는 다음과 같습니다.

```
$ pip3 install opencv-contrib-python==3.4.1.15
```

이때 지정하는 버전 번호는 Pypi 저장소에 존재하는 버전 번호를 정확히 기재해야 하는데, 저장소에서 지원하는 버전 번호 목록을 보기 위해서는 아래와 같이 버전 번호 부분을 물음표(?)로 지정하면 됩니다.

```
$ pip3 install opencv-python==?
```

어떤 모듈을 설치했든 간에 이제 파이썬 대화형 콘솔에서 cv2 모듈의 버전을 출력해 보면 정상적으로 설치되었는지 확인할 수 있습니다.

```
>>> import cv2
>>> cv2.__version__
'3.4.1'
```

1.5 라즈베리파이에서 OpenCV-Python 설치

이 책을 집필하고 있는 2019년 1월까지 OpenCV 공식 사이트와 Pypi 저장소 모두 ARM 아키텍처에서 동작하는 리눅스를 지원하는 OpenCV 바이너리 파일을 제공하고 있지 않습니다. 그래서 현재는 라즈베리파이에 OpenCV-Python을 설치하는 방법은 세 가지뿐입니다.

데비안 패키지 관리자인 APT를 이용하는 방법과 소스 코드를 직접 빌드하는 방법, 그리고 필자가 제공하는 데비안 패키지 설치 파일로 설치하는 방법입니다.

설치 방법	파이썬 버전	OpenCV 버전	OpenCV 패키지
APT 이용	2.7	2.4	메인 모듈만
소스 코드 빌드	2.7, 3.4+	모든 버전	메인, 엑스트라
패키지 설치 파일 이용	2.7, 3.4+	3.3+	메인 + 엑스트라

[표 1-3] 라즈베리파이에서 OpenCV-Python 설치 방법

표에 나와 있는 각 설치 방법 중에 원하는 방법을 한 가지 선택해서 진행하면 됩니다. 가장 빠르고 쉬운 설치 방법은 세 번째 방법인 데비안 패키지 설치 파일로 설치하는 것이며, 이 방법을 권장합니다.

1.5.1 APT로 OpenCV-Python 설치

라즈베리파이 APT 관리자로 OpenCV Python을 설치할 수 있지만, 이 방법으로는 파이썬 2.7에만 설치할 수 있고 파이썬 3에는 설치되지 않습니다. 또한, OpenCV도 2.4 버전만 설치할 수 있으며, 메인 패키지만 지원하고 엑스트라(contrib) 모듈은 포함하지 않습니다. 파이썬 3와 OpenCV 3.4를 사용하려면 이 방법은 포기해야 합니다. 이 책의 실습 예제도 실행할 수 없습니다.

APT로 설치할 패키지 이름은 `python-opencv`입니다. 설치하기 전에 패키지 목록을 새로 업데이트하고 설치합니다.

```
$ sudo apt-get update
$ sudo apt-get install python-opencv
```

파이썬 2 대화형 콘솔에서 `cv2` 모듈의 버전을 확인해 보면 설치되었는지 확인할 수 있습니다.

2장

o p e n c v

기본 입출력

이 장에서는 OpenCV로 영상을 읽고 쓰는 방법과 마우스와 키보드 이벤트를 처리하는 방법 그리고 여러 가지 선과 도형 등을 그리는 방법을 알아봅니다. 이 장의 내용은 아주 중요하지는 않지만, OpenCV로 프로그램을 작성할 때 많이 사용하는 코드이니 잘 익혀두기 바랍니다.

2.1 이미지와 비디오 입출력

OpenCV를 이용한 대부분의 작업은 파일로 된 이미지를 읽어서 적절한 연산을 적용하고 그 결과를 화면에 표시하거나 다른 파일로 저장하는 것입니다. 연산 과정에 대해서는 앞으로 자세히 알아볼 것이므로 여기서는 이미지 파일을 읽고 화면에 표시하고 저장하는 방법만 중점적으로 알아봅니다.

2.1.1 이미지 읽기

OpenCV를 사용해서 이미지를 읽고 화면에 표시하는 가장 간단한 코드는 아래와 같습니다.

[예제 2-1] 이미지 파일을 화면에 표시(img_show.py)

```
import cv2

img_file = "../img/girl.jpg" # 표시할 이미지 경로 ---①
img = cv2.imread(img_file)   # 이미지를 읽어서 img 변수에 할당 ---②

if img is not None:
    cv2.imshow('IMG', img)   # 읽은 이미지를 화면에 표시 ---③
    cv2.waitKey()           # 키가 입력될 때까지 대기 ---④
    cv2.destroyAllWindows()  # 창 모두 닫기 ---⑤
else:
    print('No image file.')
```



[그림 2-1] [예제 2-1]의 실행 결과

코드 ①의 경로에 표시할 이미지 파일이 저장되어 있어야 합니다. 이 파일을 코드 ②에서 `cv2.imread()` 함수로 읽어들입니다. 이 함수가 반환하는 타입은 다음 장에서 다룰 NumPy 배열입니다. 이 반환 값이 정상인지 아닌지 확인하고 나서 코드 ③에서 `cv2.imshow()` 함수를 써서 화면에 표시합니다. 이미지와 함께 전달한 문자열 'IMG'는 창의 제목줄에 나타납니다.

만약 코드가 코드 ③까지만 작성되어 있다면 더 이상 실행할 코드가 없어서 프로그램은 바로 종료될 것입니다. 그렇게 되면 사진을 표시한 이 창은 아주 짧은 시간 동안만 나타나 우리 눈으로는 볼 수 없게 됩니다. 그래서 코드 ④가 필요합니다. `cv2.waitKey()` 함수는 키보드의 입력이 있을 때까지 프로그램을 기다리게 합니다. 키가 입력되면 코드는 코드 ⑤의 `cv2.destroyAllWindows()` 함수에 의해서 표시한 창을 모두 닫고 나서 프로그램을 종료합니다.

[예제 2-1]에서 사용한 함수는 다음과 같습니다.

- `img = cv2.imread(file_name [, mode_flag]):` 파일로부터 이미지 읽기
 - `file_name:` 이미지 경로, 문자열
 - `mode_flag=cv2.IMREAD_COLOR:` 읽기 모드 지정
 - `cv2.IMREAD_COLOR:` 컬러(BGR) 스케일로 읽기, 기본 값
 - `cv2.IMREAD_UNCHANGED:` 파일 그대로 읽기
 - `cv2.IMREAD_GRAYSCALE:` 그레이(흑백) 스케일로 읽기

- img: 읽은 이미지, NumPy 배열
- cv2.imshow(title, img): 이미지를 화면에 표시
- title: 창 제목, 문자열
- img: 표시할 이미지, NumPy 배열
- key = cv2.waitKey([delay]): 키보드 입력 대기
 - delay=0: 키보드 입력을 대기할 시간(ms), 0: 무한대(기본 값)
 - key: 사용자가 입력한 키 값, 정수
 - -1: 대기시간 동안 키 입력 없음

cv2.imread() 함수는 파일로부터 이미지를 읽을 때 모드를 지정할 수 있습니다. 별도로 모드를 지정하지 않으면 3개 채널(B, G, R)로 구성된 컬러 스케일로 읽어들이지만, 필요에 따라 그레이 스케일 또는 파일에 저장된 스케일 그대로 읽을 수 있습니다.

```
img = cv2.imread(file_name, cv2.IMREAD_GRAYSCALE)
```

위의 코드와 같이 읽기 모드를 그레이 스케일로 지정하면 원래의 파일이 컬러 이미지일지라도 그레이 스케일로 읽습니다. 물론 그레이 이미지 파일을 cv2.IMREAD_COLOR 옵션을 지정해서 읽는다고 컬러 이미지로 읽어올 수 있는 것은 아닙니다. 이와 관련해서는 4.2절 “컬러 스페이스”에서 자세히 다룹니다.

[예제 2-2] 이미지 파일을 그레이 스케일로 화면에 표시(img_show_gray.py)

```
import cv2

img_file = "../img/girl.jpg"
img = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE) # 그레이 스케일로 읽기

if img is not None:
    cv2.imshow('IMG', img)
    cv2.waitKey()
    cv2.destroyAllWindows()
else:
    print('No image file.')
```



[그림 2-2] [예제 2-2]의 실행 결과

2.1.2 이미지 저장하기

OpenCV로 읽어들인 이미지를 다시 파일로 저장하는 함수는 `cv2.imwrite()`입니다.

- `cv2.imwrite(file_path, img)`: 이미지를 파일에 저장
 - `file_path`: 저장할 파일 경로 이름, 문자열
 - `img`: 저장할 영상, NumPy 배열

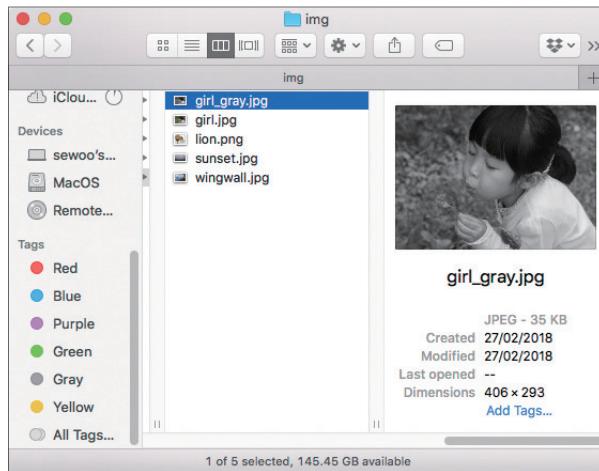
[예제 2-3]은 컬러 이미지 파일을 그레이 스케일로 읽어들여서 파일로 저장하는 예제입니다. 탐색기나 파일 등과 같은 파일 관리자로 해당 경로를 살펴보면 그레이 스케일로 바뀐 새로운 파일이 저장된 것을 확인할 수 있습니다. 저장하는 이미지의 파일 포맷은 지정한 파일 이름의 확장자에 따라서 알아서 바뀝니다.

[예제 2-3] 컬러 이미지를 그레이 스케일로 저장(img_write.py)

```
import cv2

img_file = '../img/girl.jpg'
save_file = '../img/girl_gray.jpg'

img = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE)
cv2.imshow(img_file, img)
cv2.imwrite(save_file, img) # 파일로 저장, 포맷은 확장자에 따름
cv2.waitKey()
cv2.destroyAllWindows()
```



[그림 2-3] [예제 2-3]의 실행 결과

2.1.3 동영상 및 카메라 프레임 읽기

OpenCV는 동영상 파일이나 컴퓨터에 연결한 카메라 장치로부터 연속된 이미지 프레임을 읽을 수 있는 API를 제공합니다.

다음은 동영상 파일이나 연속된 이미지 프레임을 읽을 수 있는 API의 주요 내용입니다.

- `cap = cv2.VideoCapture(file_path 또는 index)`: 비디오 캡처 객체 생성자
 - `file_path`: 동영상 파일 경로
 - `index`: 카메라 장치 번호, 0부터 순차적으로 증가(0, 1, 2, ...)
 - `cap`: `VideoCapture` 객체
- `ret = cap.isOpened()`: 객체 초기화 확인
 - `ret`: 초기화 여부, True/False
- `ret, img = cap.read()`: 영상 프레임 읽기
 - `ret`: 프레임 읽기 성공 또는 실패 여부, True/False
 - `img`: 프레임 이미지, NumPy 배열 또는 None
- `cap.set(id, value)`: 프로퍼티 변경
- `cap.get(id)`: 프로퍼티 확인
- `cap.release()`: 캡처 자원 반납

라즈베리파이 카메라 연결

라즈베리파이의 경우 카메라를 연결할 수 있는 방법이 두 가지입니다. 하나는 일반적인 웹캠을 USB에 연결하는 것이고, 나머지는 라즈베리파이 전용 카메라 장치인 파이 카메라를 CSI 슬롯에 연결하는 것입니다.

USB 형식의 웹캠은 UVC(USB Video device Class)를 지원하는 제품인 경우 USB 포트에 연결해서 바로 사용하면 됩니다.

하지만, 파이 카메라의 경우 리눅스 표준 인터페이스가 아니므로 두 가지 작업이 필요합니다.

1. 카메라 장치 활성화:

```
$ sudo raspi-config 명령에서 camera 옵션을 찾아 enable을 선택합니다.
```

2. Video4Linux 디바이스 드라이버 모듈 로딩

```
$ sudo apt-get install v4l-utils  
$ sudo modprobe bcm2835-v4l2
```

위의 과정은 부팅할 때마다 해줘야 하므로 /etc/modules 파일 마지막 라인에 bcm2835-v4l2 를 추가해 놓는 것이 좋습니다.

동영상 파일이나 컴퓨터에 연결한 카메라 장치로부터 영상 프레임을 읽기 위해서는 cv2.VideoCapture() 생성자 함수를 사용하여 객체를 생성해야 합니다. 이 함수에 동영상 파일 경로 이름을 전달하면 동영상 파일에 저장된 프레임을 읽을 수 있고, 카메라 장치 번호를 전달하면 카메라로 촬영하는 프레임을 읽을 수 있습니다.

객체를 생성하고 나면 isOpened() 함수로 파일이나 카메라 장치에 제대로 연결되었는지 확인할 수 있고, 연결이 잘 되었다면 read() 함수로 다음 프레임을 읽을 수 있습니다. read() 함수는 Boolean과 NumPy 배열 객체를 쌍으로 갖는 튜플 (ret, img) 객체를 반환하는데, 다음 프레임을 제대로 읽었는지에 따라 ret 값이 정해집니다. 만약 ret 값이 True이면 다음 프레임 읽기에 성공한 것이고, img를 꺼내서 사용하면 됩니다. 만약 ret 값이 False이면 다음 프레임 읽기에 실패한 것이고, 튜플의 나머지 값인 img는 None입니다. 다음 프레임 읽기에 실패하는 경우는 파일이나 장치에 문제가 있거나 파일의 끝에 도달했을 경우입니다.

비디오 캡처 객체의 set(), get() 함수를 이용하면 여러 가지 속성을 얻거나 지정할 수 있으며, 프로그램을 종료하기 전에 release() 함수를 호출해서 자원을 반납해야 합니다.

2.1.4 동영상 파일 읽기

[예제 2-4]는 동영상 파일을 읽기 위한 간단한 코드입니다.

[예제 2-4] 동영상 파일 재생(video_play.py)

```
import cv2

video_file = "../img/big_buck.avi" # 동영상 파일 경로

cap = cv2.VideoCapture(video_file) # 동영상 캡처 객체 생성 ---①
if cap.isOpened(): # 캡처 객체 초기화 확인
    while True:
        ret, img = cap.read() # 다음 프레임 읽기 ---②
        if ret: # 프레임 읽기 정상
            cv2.imshow(video_file, img) # 화면에 표시 ---③
            cv2.waitKey(25) # 25ms 지연(40fps로 가정) ---④
        else: # 다음 프레임을 읽을 수 없음
            break # 재생 완료
    else:
        print("can't open video.") # 캡처 객체 초기화 실패
cap.release() # 캡처 자원 반납
cv2.destroyAllWindows()
```



[그림 2-4] [예제 2-4]의 실행 결과¹

¹ [예제 2-4]에서 사용한 동영상은 <http://bbb3d.renderfarming.net/download.html>에서 다운로드할 수 있습니다.

코드 ①에서는 `cv2.VideoCapture()` 함수에 동영상 파일 경로를 전달해서 캡처 객체 `cap`을 생성합니다. 캡처 객체가 정상적으로 지정한 파일로 초기화되면 `cap.isOpened()` 함수는 `True`를 반환합니다. 연속해서 파일의 프레임을 읽어오기 위해서 무한 루프를 돌리면서 `cap.read()`를 호출하는데, 이 함수는 정상적인 프레임 읽기가 되었는지를 확인할 수 있는 불(boolean) 변수와 한 개의 프레임 이미지를 표현한 NumPy 배열 객체를 쌍으로 갖는 튜플 객체를 반환합니다. 그 다음 프레임 이미지를 화면에 표시하는 것은 이전의 코드와 거의 비슷합니다.

코드 ④에서 `cv2.waitKey(25)`가 필요한 이유는 각 프레임을 화면에 표시하는 시간이 너무 빠르면 우리 눈으로 볼 수 없기 때문입니다. 이때 자연 시간은 실습에 사용할 동영상의 FPS(Frames Per Second, 초당 프레임 수)에 맞게 조정해서 적절한 속도로 영상을 재생하게 해야 합니다. 보통 동영상 파일이 40fps인 경우가 가장 많아 [예제 2-4]에서는 25ms의 자연 시간을 사용했습니다.

FPS와 자연 시간 구하기

동영상 파일의 정확한 FPS를 쉽게 얻는 방법은 곰플레이어, 다음팟플레이어, VLC 등과 같은 무료 동영상 플레이어에서 속성 값을 확인하는 것입니다. OpenCV로 FPS를 구하는 방법은 앞으로 2.1.6절 “카메라 비디오 속성 제어”에서 다시 다루겠습니다. FPS를 대충 추정하거나 다른 플레이어로 구했다면 이에 맞는 자연 시간을 구해야 할 것입니다. FPS에 맞는 자연 시간을 구하는 공식은 1초에 몇 개의 사진이 들어가야 하는가를 구하는 것으로 다음과 같습니다.

$$\text{지연시간} = 1000 \div \text{fps}$$

1,000으로 계산하는 이유는 1초를 밀리초(ms) 단위로 환산해서 제공해야 하기 때문입니다. FPS를 40으로 가정해서 대입한 결과는 다음과 같습니다.

$$25 = 1000 \div 40$$

2.1.5 카메라(웹캠) 프레임 읽기

카메라로 프레임을 읽기 위해서는 `cv2.VideoCapture()` 함수에 동영상 파일 경로 대신에 카메라 장치 인덱스 번호를 정수로 지정해 주면 됩니다. 카메라 장치 인덱스 번호는 0부터 시작해서 1씩 증가합니다. 만약 카메라가 하나만 연결되어 있으면 당연히 0번 인덱스를 사용하면 됩니다. 이 부분을 제외하고는 나머지 코드는 동영상 파일을 읽는 것과 거의 똑같습니다.

[예제 2-5] 카메라 프레임 읽기(video_cam.py)

```

import cv2

cap = cv2.VideoCapture(0)                      # 0번 카메라 장치 연결 ---①
if cap.isOpened():
    while True:
        ret, img = cap.read()                  # 카메라 프레임 읽기
        if ret:
            cv2.imshow('camera', img)          # 프레임 이미지 표시
            if cv2.waitKey(1) != -1:           # 1ms 동안 키 입력 대기 ---②
                break                         # 아무 키나 눌렀으면 종지
        else:
            print('no frame')
            break
    else:
        print("can't open camera.")
cap.release()
cv2.destroyAllWindows()

```



[그림 2-5] [예제 2-5]의 실행 결과

코드 ①에서는 0번 카메라 장치에서 촬영한 프레임을 읽어서 화면에 표시합니다. 동영상 파일과는 다르게 카메라로부터 프레임을 읽는 경우 파일의 끝이 정해져 있지 않으므로 무한루프를 빠져 나올 조건이 없습니다. 그래서 코드 ②에서 사용자가 아무 키나 누르면 빠져 나오게 했습니다. 따라서 이 프로그램을 종료하려면 키보드의 아무 키나 누르면 됩니다. `cv2.waitKey()` 함수는 지정한 대기 시간 동안 키 입력이