x

K-Digital Training 스마트 팩토리 3기

OpenCV

OpenCV란?

- https://opencv.org/
- Open Source Computer Vision
- 영상 처리에 사용할 수 있는 오픈 소스 라이브러리
- 인텔에서 C/C++ 언어로 개발
- 공장에서 제품 검사, 의료 영상 처리 및 보정, CCTV, 로보틱스 등에 활용
- 오픈 소스이면서 BSD 라이선스를 따르기 때문에 상업적 목적으로 사용 가능
- C, C++, Python, Java 등 지원
- https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html

OpenCV란?

- 기본적으로 numpy 배열 이용
- (width, height, channel)
- RGB가 아닌 BGR 이용

설치

- 설치
  - pip install opencv-python
- Import
  - import cv2

윈도우 만들기

- cv2.namedWindow(winname[,flags= WINDOW_AUTOSIZE | WINDOW_KEEPRATIO | WINDOW_GUI_EXPANDED])
  - 윈도우 생성
  - winname : 윈도우 이름
  - flags : 옵션, 윈도우를 만들 때 적용할 초기 상태
    - WINDOW_NORMAL : 유저가 윈도우 크기 변경 가능, 전체화면 변경 가능
    - WINDOW_AUTOSIZE : 유저가 윈도우 크기 변경 불가
    - WINDOW_OPENGL : OPENGL 지원하는 윈도우
    - WINDOW_FULLSCREEN : 전체화면 윈도우
    - WINDOW_FREERATIO : 이미지의 비율 상관없이 확장됨
    - WINDOW_KEEPRATIO : 이미지의 비율 고정
    - WINDOW_GUI_EXPANDED : status bar와 tool bar 존재
    - WINDOW_GUI_NORMAL : 예전 방식

이미지 입력

- cv2.imread(path[,flags= IMREAD_ANYCOLOR]) -> retval
  - 이미지 읽기
  - path : 이미지 파일의 상대 경로, 절대 경로
  - flags : 옵션, 이미지를 불러올 때 적용할 초기 상태
    - IMREAD_ANYCOLOR : default, 가능한 3채널 이미지
    - IMREAD_COLOR : 3채널 BGR 이미지
    - IMREAD_UNCHANGED : 원본 그대로, Exif orientation 무시
    - IMREAD_GRAYSCALE : 채널이 하나인 회색 이미지
    - 등등…

이미지 출력

- cv2.imshow(winname, mat)
  - 이미지 출력
  - winname : 윈도우 이름
  - mat : 보여질 이미지의 배열

키보드 입력

- cv2.waitKey([,delay=0]) -> retval
  - 키보드 입력을 기다림
  - 누른 키의 코드를 반환하거나 지정된 시간이 경과하기 전에 아무키도 누르지 않은 경우 -1을 반환
  - 키 누르기를 기다리지 않으려면 pollKey()를 사용
  - 0은 '영원히'를 의미하는 특별한 값

키보드 입력

- ord()
  - 문자의 아스키 코드 획득
- chr()
  - 아스키 코드를 문자로 변경

```
key = cv2.waitKey(0)
if key == ord('q') :
  print(chr(key))
```

창 닫기

- cv2.destroyAllWinows()
  - 모든 창을 닫음

- cv2.destroyWinow(winname)
  - 지정된 창을 닫음
  - winname : 닫을 창의 이름

영상 입력 초기화

- cv2.VideoCapture(int index, apiPreference= CAP_ANY) -> VideoCapture
  - 카메라 읽기
  - index : 비디오를 읽어들일 장치의 id, 기본 카메라의 경우는 0
  - apiPreference : 옵션, 사용할 캡쳐 API backend
  - VideoCapture 클래스 반환

- cv2.VideoCapture(String filename, apiPreference= CAP_ANY) -> VideoCapture
  - 비디오 읽기
  - filename : 비디오 파일의 이름, Url or 이미지의 배열 or Gstreamer pipeline string
  - apiPreference : 옵션, 사용할 캡쳐 API backend
  - VideoCapture 클래스 반환

영상 입력 설정

- VideoCapture.set(int propId, double value) -> retval
  - 캡쳐 속성 설정
  - propId : 속성 Id
    - CAP_PROP_POS_MSEC : msec 단위의 위치
    - CAP_PROP_POS_FRAMES : frame 단위의 위치
    - CAP_PROP_FRAME_WIDTH : 프레임의 넓이
    - CAP_PROP_FRAME_HEIGHT : 프레임의 높이
    - CAP_PROP_FPS : 프레임 레이트
    - CAP_PROP_FRAME_COUNT : 총 프레임 개수
    - CAP_PROP_BRIGHTNESS : 이미지의 밝기 (카메라가 지원할 경우에)
    - 등등…
  - value : 설정할 속성 값

영상 정보 얻기

- VideoCapture.get(int propId) -> retval
  - 영상 속성 얻기
  - propId : 속성 Id
  - retvalue : 얻어온 속성 값

영상 입력

- VideoCapture.read([,image]) -> retval, image
  - 영상 읽기
  - retval : false – 만약에 출력할 프레임이 없다면
  - [out] image : 출력할 이미지 배열
  - grap()(다음 프레임을 캡쳐) 과 retrieve()(디코드)의 함수를 묶은 함수

영상 입력 종료

- VideoCapture.release()
  - 비디오 파일이나 입력 디바이스 닫기
  - VideoCapture 클래스가 open되면 소멸자에서 자동으로 호출

이미지 조정

- cv2.flip(input, flipCode[, dst]) -> dst
  - input : 원본 이미지
  - dst : 원본과 동일한 크기의 이미지
  - flipCode : 어떻게 flip 시킬지
    - = 0 : x축 반전
    - > 0 : y축 반전
    - < 0 : x,y축 반전

이미지 조정

- cv2. getRotationMatrix2D(center, angle, scale) -> retval
  - center : 회전 중심점
  - angle : 회전 시킬 각도(반시계방향)
  - scale : 크기 factor

이미지 조정

- cv2.wrapAffine(src, matrix, (width, height)[,dst[,flags[,borderMode[,borderValue]]]]) -> dst
  - 아핀 변환 함수 적용
  - src : 원본 이미지
  - matrix : 아핀 맵 행렬
  - size : 출력 이미지 크기
  - dst : 출력 이미지
  - flags : 보간 방법과 option flag의 조합
  - borderMode : 픽셀 외삽 방법
  - borderValue : 일정한 경계의 경우 사용되는 값, 기본적으로 0

이미지 조정

- cv2. pyrUp(src[, dst[, dsize[, borderType]]]) -> dst
  - 이미지 크기를 키우는 함수
  - src : 입력 이미지
  - dst : 결과 이미지
  - dsize : 결과 이미지의 크기
  - borderType : 경계 type

- cv2. pyrDown(src[, dst[, dsize[, borderType]]]) -> dst
  - 이미지 크기를 줄이는 함수

- 기본적으로 2배씩 확대, 축소

이미지 조정

- borderType : 테두리 보정 방법

Enumerator	
BORDER_CONSTANT Python: cv.BORDER_CONSTANT	`iiiiii|abcdefgh|iiiiiii` with some specified `i`
BORDER_REPLICATE Python: cv.BORDER_REPLICATE	`aaaaaa|abcdefgh|hhhhhhh`
BORDER_REFLECT Python: cv.BORDER_REFLECT	`fedcba|abcdefgh|hgfedcb`
BORDER_WRAP Python: cv.BORDER_WRAP	`cdefgh|abcdefgh|abcdefg`
BORDER_REFLECT_101 Python: cv.BORDER_REFLECT_101	`gfedcb|abcdefgh|gfedcba`
BORDER_TRANSPARENT Python: cv.BORDER_TRANSPARENT	`uvwxyz|abcdefgh|ijklmno`
BORDER_REFLECT101 Python: cv.BORDER_REFLECT101	same as BORDER_REFLECT_101
BORDER_DEFAULT Python: cv.BORDER_DEFAULT	same as BORDER_REFLECT_101
BORDER_ISOLATED Python: cv.BORDER_ISOLATED	do not look outside of ROI

이미지 조정

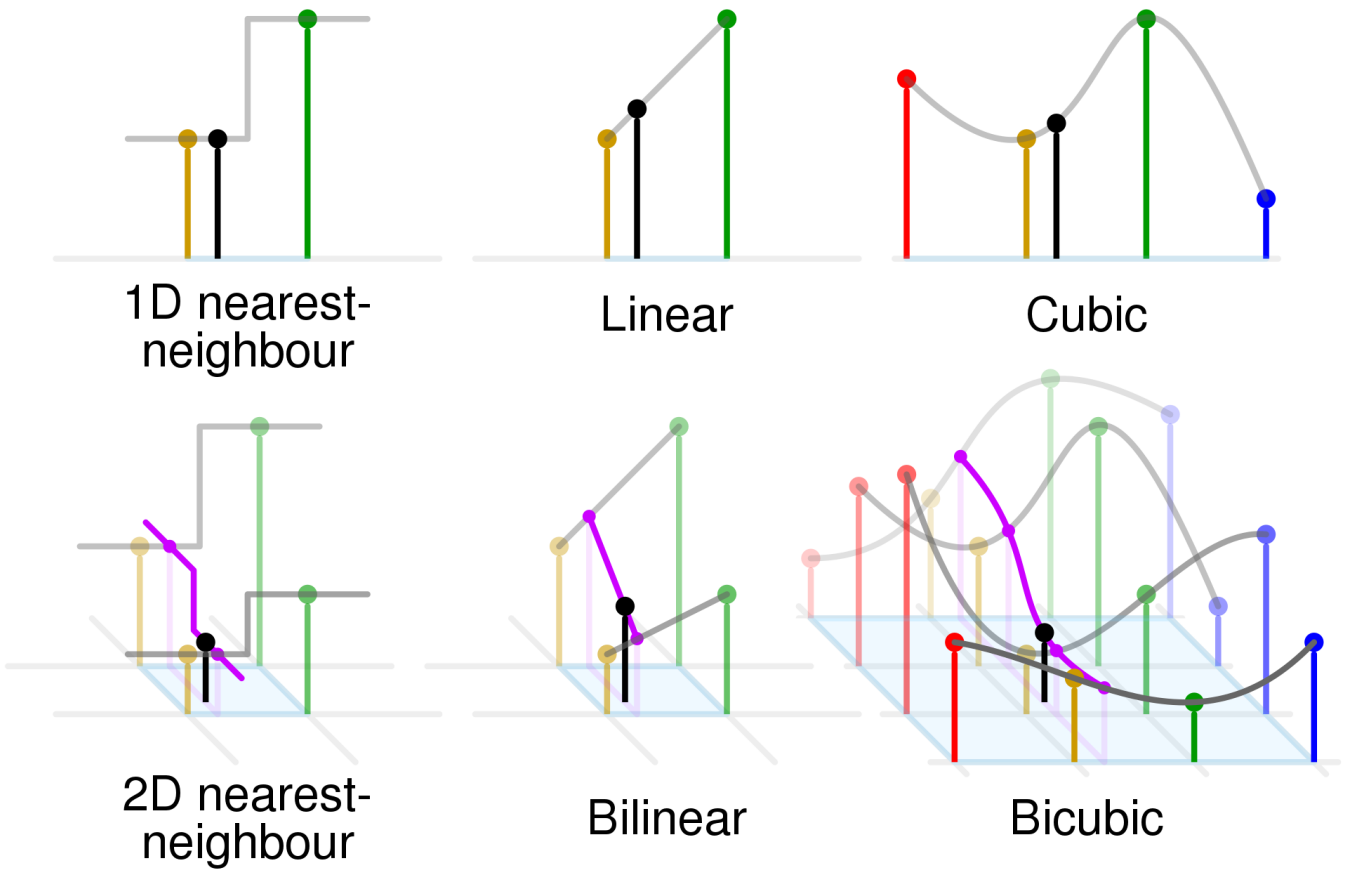
- cv2.resize(src, dsize[, dst[, fx=0[, fy=0[,
  interpolation=INTER_LINEAR]]]]) -> dst
  - 이미지 크기 조정 함수
  - src : 입력 이미지
  - dst : 출력 이미지
  - dsize : 출력 이미지 크기
  - fx : 가로축에 대한 scale factor
  - fy : 세로축에 대한 scale factor
  - dsize 또는 fx, fy 는 모두 0이 아니어야함
  - interpolation : 보간법

이미지 조정

- interpolation (보간법) 종류
  - INTER_NEAREST : 최근방 이웃 보간법
  - INTER_LINEAR : 양선형 보간법(가장 많이 씀) (2x2 이웃 픽셀 참조)
  - INTER_CUBIC : 3차회선 보간법 (4x4 이웃 픽셀 참조)
  - INTER_LANCZOS4 : 랑코즈 보간법 (8x8 이웃 픽셀 참조)
  - INTER_AREA : 영상 축소시 주로 사용

이미지 조정

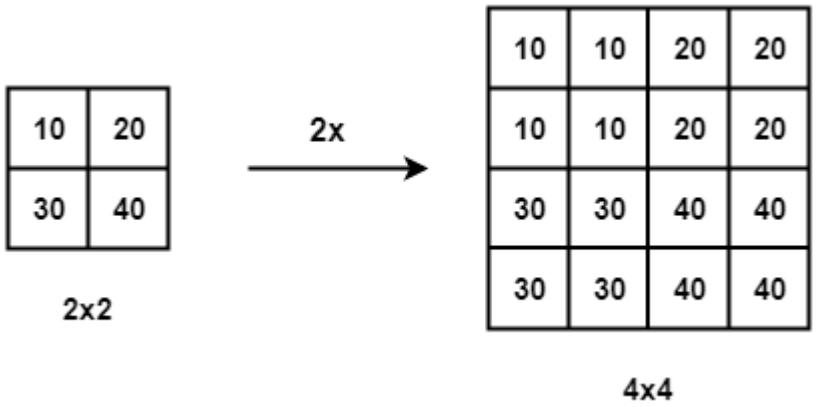
- interpolation (보간법) 종류



이미지 조정

- interpolation (보간법) 종류

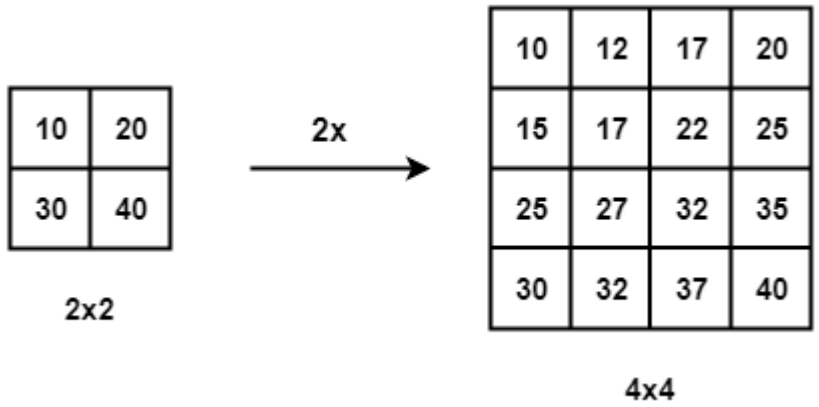
nearest



2x2

4x4

bilinear



2x2

4x4

이미지 조정

- 이미지 slice
  - dst = src : 얕은 복사, dst가 바뀌면 src도 영향을 받음
  - dst = src.copy() : 깊은 복사

- dst = src[100:200, 200:300]
- dst = src[100:200, 200:300].copy()
- dst[0:300, 0:300] = src[100:400, 200:500]

이미지 조정

- src.shape
  - (가로, 세로, 채널) 정보

이미지 조정

- cv2.cvtColor(src, code[,dst[, dstCn=0]]) -> dst
  - 이미지 color space 변경
  - src : 입력 이미지
  - dst : 출력 이미지
  - code : 변경할 color space code
    - 참고 https://docs.opencv.org/4.x/d8/d01/group__imgproc__color__conversions.html#ga4e0972be5de079fed4e3a10e24ef5ef0
  - dstCn : 변경할 이미지의 채널 개수

이미지 조정

- cv2.split(m[,mv]) -> mv
  - 이미지 채널 분리
  - m : 입력 이미지 (multi channel array)
  - mv : 출력 어레이

- cv2.merge(mv[,dst]) -> dst
  - 이미지 채널 병합
  - mv : 병합할 매트릭스, size와 depth가 동일해야함
  - dst : 입력과 depth가 같은 출력 어레이

- OpenCV는 RGB가 아닌 BGR 포멧을 기본으로 사용

이미지 조정

- cv2.threshold(src, thresh, maxval, type) -> retval, dst
  - array 에 threshold 적용
  - src : 입력 배열
  - dst : 출력 배열
  - thresh : threshold 값
  - maxval : threshold 타입에서에 최대값
  - type : thresholding type

이미지 조정

Enumerator	
THRESH_BINARY Python: cv.THRESH_BINARY	$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV Python: cv.THRESH_BINARY_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC Python: cv.THRESH_TRUNC	$\texttt{dst}(x,y) = \begin{cases} \texttt{threshold} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_TOZERO Python: cv.THRESH_TOZERO	$\texttt{dst}(x,y) = \begin{cases} \texttt{src}(x,y) & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV Python: cv.THRESH_TOZERO_INV	$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{src}(x,y) & \text{otherwise} \end{cases}$
THRESH_MASK Python: cv.THRESH_MASK	
THRESH_OTSU Python: cv.THRESH_OTSU	flag, use Otsu algorithm to choose the optimal threshold value
THRESH_TRIANGLE Python: cv.THRESH_TRIANGLE	flag, use Triangle algorithm to choose the optimal threshold value