배열 연산 함수

기본 배열 처리 함수

❖ 영상 속성과 픽셀 접근

- ✓ OpenCV 파이썬은 영상을 numpy.ndarray을 이용하여 표현
- ✓ 영상의 중요 속성(shape, dtype)을 확인하고 numpy의 astype(), reshape()로 속성을 변경하고 영상 픽셀를 y(행), x(열) 순으로 인덱스를 지정한 후 접근
- ✓ numpy는 다중 채널 영상을 모양(shape)에 의해 표현하고 OpenCV는 픽셀 자료형으로 표현

구분	numpy 자료형	OpenCV 자료형, 1-채널
8비트 unsigned 정수	np.uint8	cv2.CV_8U
8비트 signed 정수	np.int8	cv2.CV_8S
16비트 unsigned 정수	np.uint16	cv2.CV_16U
16비트 signed 정수	np.int16	cv2.CV_16S
32비트 signed 정수	np.int32	cv2.CV_32S
32비트 실수	np.float32	cv2.CV_32F
64비트 실수	np.float64	cv2.CV_64F

기본 배열 처리 함수

- ❖ 영상 속성과 픽셀 접근
  - ✓ 이미지 크기 변경
    - ❑ resize 함수를 이용해서 이미지 크기 변경

      resize(이미지 데이터, (가로 크기, 세로 크기))
    - ❑ 이미지들은 제각기 다양한 크기를 가지는데 이를 특성으로 사용하려면 동일한 차원으로 만들어야 하기 때문에 이미지 크기를 변경하는데 이미지는 행렬에 정보를 담고 있기 때문에 이미지 크기를 줄이면 행렬 크기와 거기에 담긴 정보도 줄어 듬
    - ❑ 머신 러닝은 수 천 에서 수 십만 개의 이미지가 필요한데 이미지가 클수록 메모리를 많이 차지하기 때문에 이미지 크기를 줄여서 메모리 사용량을 줄일 수 있음
      - ○ 머신 러닝에서 많이 사용하는 이미지의 크기는 32X32, 64X64, 96X96, 256X256 등

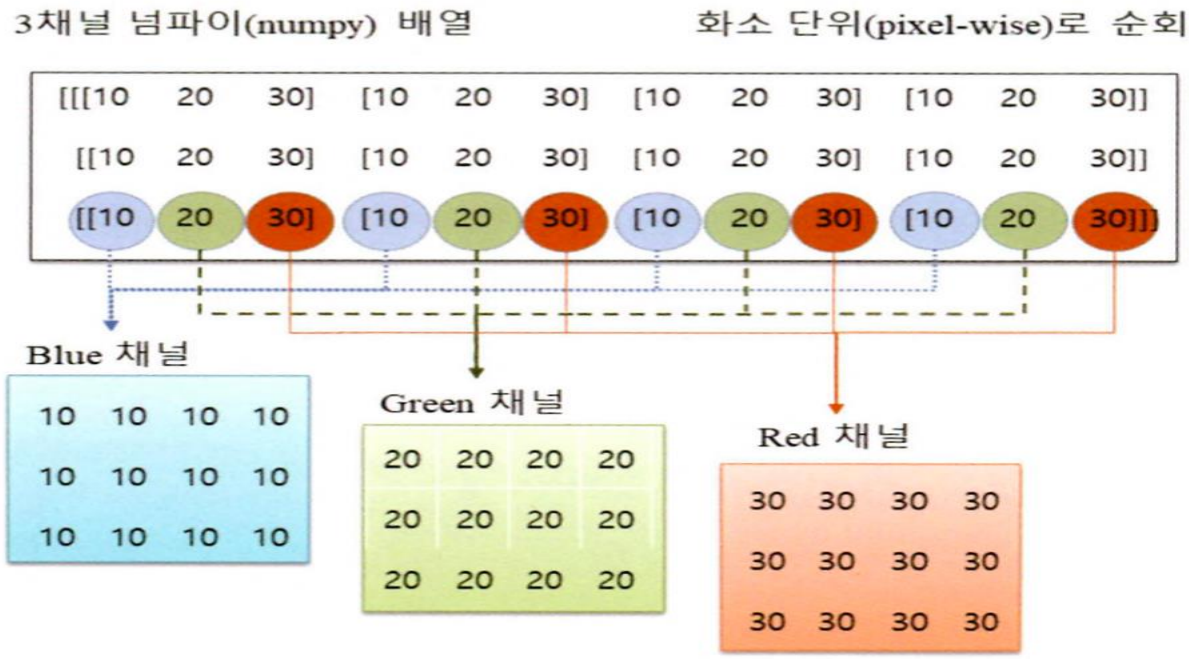
기본 배열 처리 함수

- ❖ 영상 속성과 픽셀 접근
  - ✓ 이미지 크기 변경
    - ❑ img.flatten()은 다차원 배열을 1차원 배열로 변경하여 img.shape = (262144,)
    - ❑ 3 img.reshape(-1, 512, 512)는 3차원 배열로 확장하는데 -1로 표시된 부분은 크기를 자동으로 계산하는데 img의 픽셀 크기가 512 X 512이므로 img.shape = (1, 512, 512)로 변경되고 img[0].shape은 (512, 512)
    - ❑ cv2.imshow('img', img[0])은 원본 영상을 표시
    - ❑ img.reshape()은 실제 데이터를 변경하지는 않고 모양을 변경
    - ❑ 영상의 확대 축소 크기는 cv2.resize()로 변환

채널 처리 함수

❖ 영상 채널 분리와 병합

✓ 3채널 numpy 배열 및 컬러 영상의 각 채널



✓ cv2.split()는 다중 채널 영상을 리스트에 단일 채널 영상으로 분리하고 cv2.merge()는 단일 채널 영상을 병합하여 다중 채널 영상을 생성

cv2.split(m[, mv]) -> mv

cv2.merge(mv[, dst]) -> dst

채널 처리 함수

- ❖ 영상 채널 분리와 병합
  - ✓ 채널 분리
    - ❑ dst = cv2.split(src)는 3-채널 BGR 컬러 영상 src를 채널 분리하여 리스트 dst에 저장
    - ❑ type(dst)는 'list'이고 type(dst[0]), type(dst[1]), type(dst[2])는 'numpy.ndarray'
    - ❑ 채널 순서는 0-채널은 Blue, 1-채널은 Green, 2-채널은 Red

회전 처리 함수

- ❖ 회전
  - ✓ 입력된 2차원 배열을 수직, 수평 양축으로 뒤집기

    cv2.flip(src, flipCode[, dst]) -> dst
    - ❑ src , dst: 입력 배열, 출력 배열
    - ❑ flipCode: 배열을 뒤집는 축
      - ○ 0 : X축을 기준으로 위아래로 뒤집기
      - ○ 1: y축을 기준으로 좌우로 뒤집기
      - ○ -1: 양축(x축, y축 모두)을 기준으로 뒤집기
  - ✓ 입력된 배열을 복사

    cv2.repeat(src, ny, nx[, dst) -> dst
    - ❑ src , dst: 입력 배열, 출력 배열
    - ❑ ny, nx: 수직 방향, 수평 방향 반복 횟수
  - ✓ 입력 행렬의 전치 행렬을 출력으로 리턴
    - ❑ cv2.transpose(src[, dst]) -> dst
    - ❑ src, dst: 입력 배열, 출력 배열

회전 처리 함수

- ❖ 회전
  - ✓ 상하좌우로 뒤집기

회전 처리 함수

❖ 회전

cv2.getRotationMatrix2D(center, angle, scale) -> M

✓ center 좌표를 중심으로 scale 만큼 확대/축소하고 angle 각도만큼 회전한 어파인 변환 행렬 M을 반환

✓ angle > 0 이면 반 시계 방향 회전

✓ M[:, 2] += (tx, ty)를 추가하면 이동을 추가할 수 있음

$$M = \begin{bmatrix} \alpha & \beta & (1-\alpha) \times center.x - \beta \times center.y \\ -\beta & \alpha & \beta \times center.x + (1-\alpha) \times center.y \end{bmatrix}$$

$$\alpha = scale \times cos(RADIAN(angle))$$

$$\beta = scale \times sin(RADIAN(angle))$$

회전 처리 함수

❖ 회전

cv2.warpAffine(src, M, dsize[, dst[, flags[,borderMode, borderValue]]]) -> dst

- ✓ cv2.warpAffine() 함수는 src 영상에 2 X 3 어파인 변환 행렬 M을 적용하여 dst에 반환
- ✓ dsize는 출력 영상 dst의 크기이며 flags는 보간법(cv2.INTER_NEAREST, cv2.INTER_LINEAR 등) 과 cv2_WARP_INVERSE_MAP의 조합
- ✓ Cv2.WARP_INVERSE_MAP은 이이 dst -> src의 역변환 을 의미
- ✓ borderMode는 경계값 처리 방식으로 borderMode = cv2_BORDER_CONSTANT 에서 borderValue는 경계값 상수

산술 연산 함수

- 산술 연산, 비트 연산, 비교 범위, 수치 연산 함수
  - ✓ OpenCV_Python은 영상을 numpy.ndarray로 표현하기 때문에 numpy 연산을 사용할 수 있음
  - ✓ numpy 연산을 사용할 때 연산 결과가 자료형의 범위를 벗어나는 경우 주의해서 사용해야 하는데 uint8 자료형의 영상 src1과 src2에서 dst = src1 + src2의 연산을 하는 경우 255를 넘는 픽셀 값은 256으로 나눈 나머지를 갖지만 dst = cv2.add(src1, src2)는 255를 넘는 픽셀 값은 최대값 255를 저장 – saturate()
  - ✓ 함수들의 공통 설정
    - ❑ src, src1, src2는 입력 영상이고 dst는 연산의 결과 영상
    - ❑ mask는 8-비트 1-채널 마스크 영상으로 mask(y, x) 0인 아닌 픽셀에서만 연산을 수행
    - ❑ dtype에 출력 영상의 픽셀 자료형을 명시할 경우 cv2.CV_8U, cv2.CV_8UCl, cv2. CV_8UC3, cv2.CV_8UCl, cv2.CV_16S, cv2.CV_16SCl, cv2.CV_32F, cv2. CV_32FC3, cv2.CV_64F, cv2.CV_64FC3 등과 같이 비트, 타입, 채널 수를 명시한 데이터를 사용
    - ❑ OpenCV 함수의 매개변수 dst를 생략하거나 dst = None을 사용하면 결과 영상을 생성하여 반환

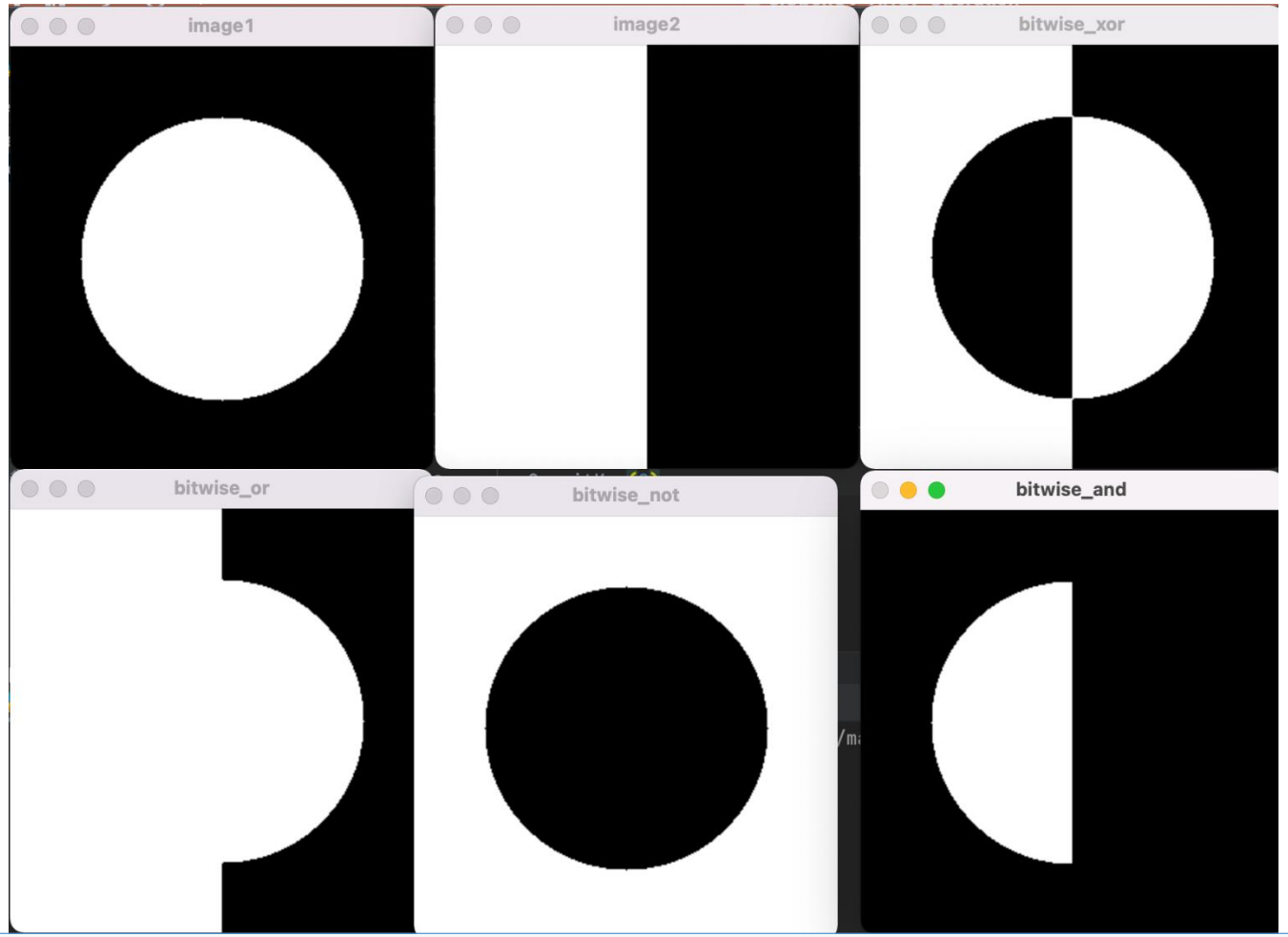
산술 연산 함수

❖ 산술 연산, 비트 연산, 비교 범위, 수치 연산 함수

사칙 연산	cv2.add(src1, src2[, dst[, mask[, dtype]]]) -> dst
	cv2.addWeighted(src1, alpha, src2, beta, gamma[, dst[, dtype]]) -> dst
	cv2.subtract(src1, src2[, dst[, mask[, dtype]]]) -> dst
	cv2.scaleAdd(src1, alpha, src2[, dst]) -> dst
	cv2.multiply(src1, src2[, dst[, scale[, dtype]]]) -> dst
	cv2.divide(src1, src2[, dst[, scale[, dtype]]]) -> dst
	cv2.divide(scale, src2[, dst[, dtype]]) -> dst
비트 연산	cv2.bitwise_not(src[, dst[, mask]]) -> dst
	cv2.bitwise_and(src1, src2[, dst[, mask]]) -> dst
	cv2.bitwise_or(src1, src2[, dst[, mask]]) -> dst
	cv2.bitwise_xor(src1, src2[, dst[, mask]]) -> dst
비교 범위 연산	cv2.compare(src1, src2, cmpop[, dst]) -> dst
	cv2.checkRange(a[, quiet[, minVal[, maxVal]]]) -> retval, pos
	cv2.inRange(src, lowerb, upperb[, dst]) -> dst
수치 연산	cv2.absdiff(src1, src2[, dst]) -> dst
	cv2.convertScaleAbs(src[, dst[, alpha[, beta]]]) -> dst
	cv2.exp(src[, dst]) -> dst
	cv2.log(src[, dst]) -> dst
	cv2.pow(src, power[, dst]) -> dst
	cv2.sqrt(src[, dst]) -> dst
	cv2.magnitude(x, y[, magnitude]) -> magnitude
	cv2.phase(x, y[, angle[, angleInDegrees]]) -> angle
	cv2.cartToPolar(x, y[, magnitude[, angle[, angleInDegrees]]]) -> magnitude, angle
	cv2.polarToCart(magnitude, angle[, x[, y[, angleInDegrees]]]) -> x, y

산술 연산 함수

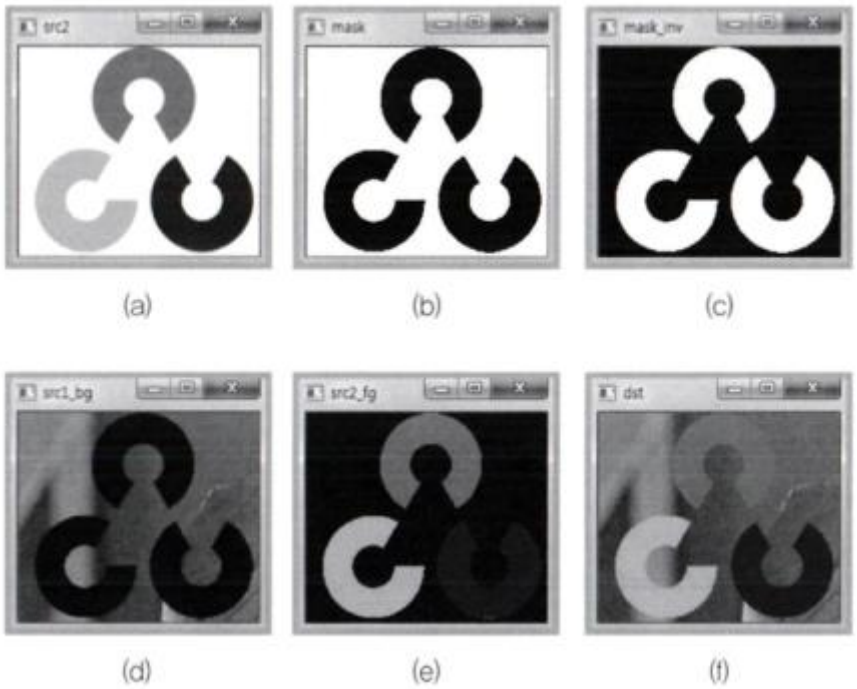
- ❖ 산술 연산, 비트 연산, 비교 범위, 수치 연산 함수
  - ✓ 비트 연산



산술 연산 함수

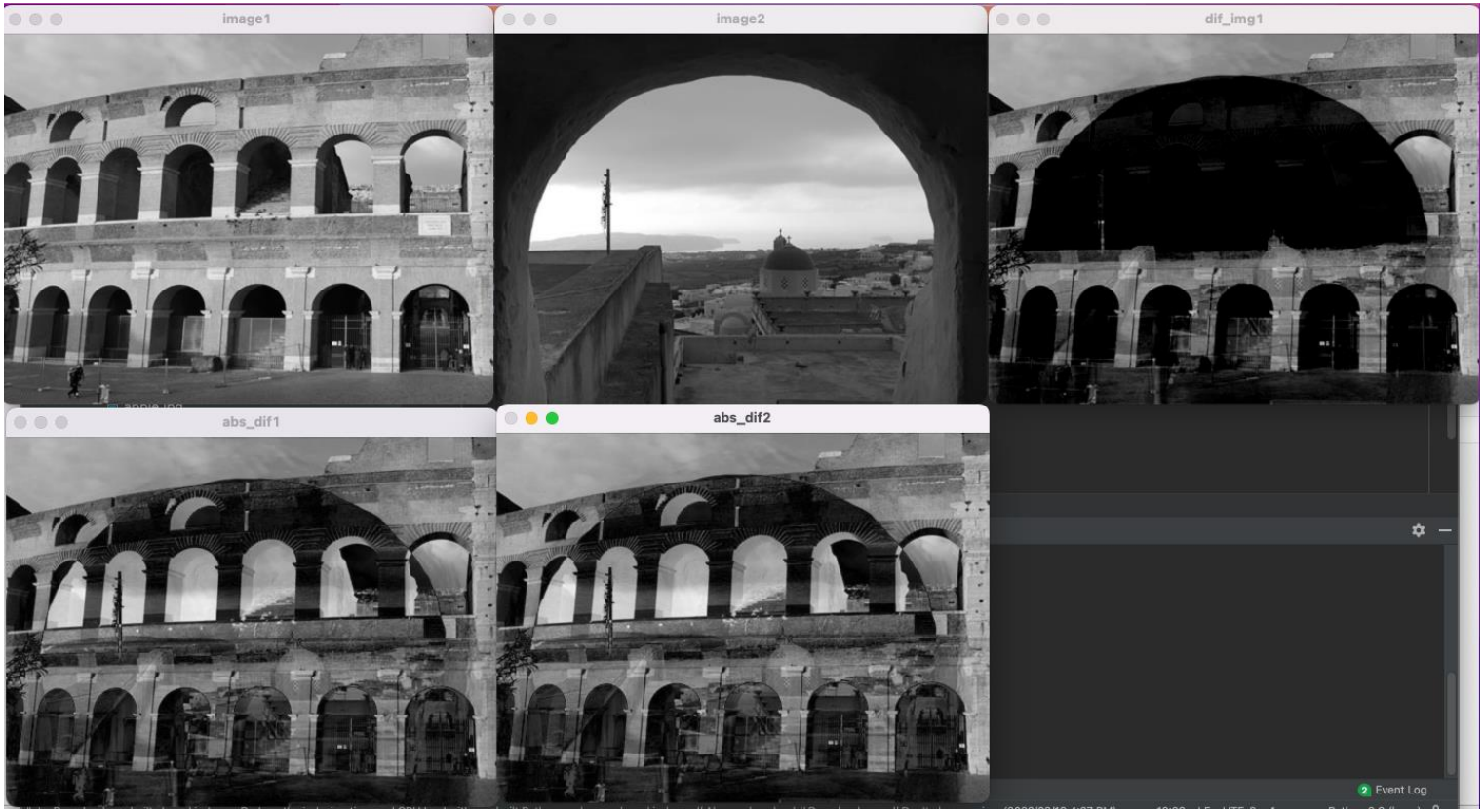
❖ 산술 연산, 비트 연산, 비교 범위, 수치 연산 함수

✓ 비트 연산



산술 연산 함수

- ❖ 산술 연산, 비트 연산, 비교 범위, 수치 연산 함수
  - ✓ 절대값 연산



통계 관련 함수

❖ 수학 및 통계 함수

정규화	cv2.norm(src1, src2[, normType[, mask]]) -> retval
	cv2.normalize(src, dst[, alpha[,beta[,norm_type[, dtype[,mask]]]]]) -> dst
최대 최소	cv2.min(src1, src2[, dst]) -> dst
	cv2.max(src1, src2[, dst]) -> dst
	cv2.minMaxLoc(src[, mask]) -> minVal, maxVal, minLoc, maxLoc
통계	cv2.countNonZero(src) -> retval
	cv2.reduce(src, dim, rtype[, dst[, dtype]]) -> dst
	cv2.mean(src[, mask]) -> retval
	cv2.meanStdDev(src[, mean[, stddev[, mask]]]) -> mean, stddev
	cv2.calcCovarMatrix(samples, flags[, covar[, mean[, ctype]]]) -> covar, mean
	cv2.Mahalanobis(v1, v2, icovar) -> retval

통계 관련 함수

❖ 수학 및 통계 함수

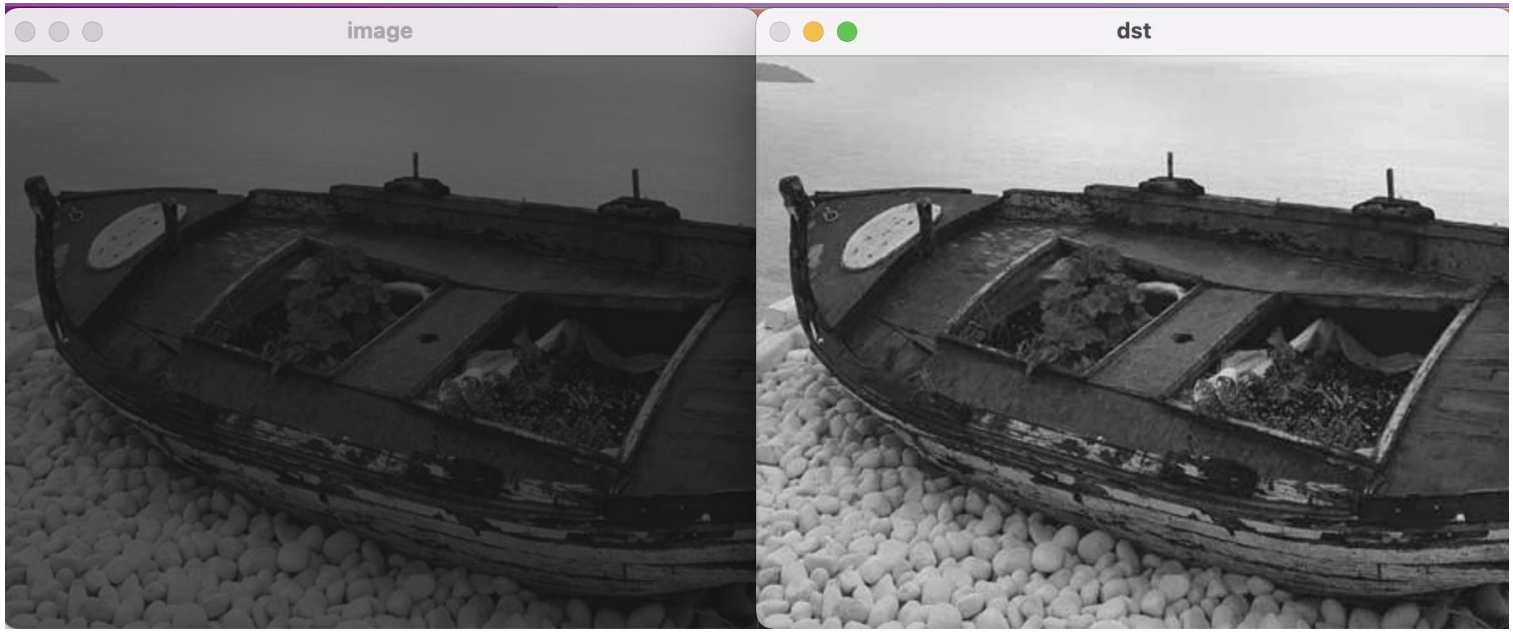
난수	cv2.randu(dst, low, high) -> dst
	cv2.randn(dst, mean, stddev) -> dst
	cv2.randShuffle(dst[, iterFactor]) -> dst
선형대수	cv2.eigen(src[, eigenvalues[, eigenvectors]]) -> retval, eigenvalues, eigenvectors
	cv2.PCACompute(data, mean[, eigenvectors[, maxComponents]]) -> mean, eigenvectors
	cv2.PCAProject(data, mean, eigenvectors[, result]) -> result
	cv2.PCABackProject(data, mean, eigenvectors[, result]) -> result
정렬	cv2.sort(src, flags[, dst]) -> dst
	cv2.sortIdx(src, flags[, dst]) -> dst

통계 관련 함수

- 수학 및 통계 함수
  - ✓ 최소값과 최대값
    - 두 행렬의 각 원소를 비교하여 큰 값이나 작은 값을 찾아야 할 때가 종종 있음
    - 하나의 행렬에서 가장 큰 값이나 가장 작은 값의 위치를 알면 히스토그램을 그리거나 다른 행렬의 연산 등에서 상당히 유용
    - 사용할 수 있는 함수는 cv2.max(), cv2.min(), cv2.minMaxLoc() 등

통계 관련 함수

- ❖ 수학 및 통계 함수
  - ✓ 최소값과 최대값을 이용한 화질 개선 – 차이가 적은 경우



통계 관련 함수

- ❖ 수학 및 통계 함수
  - ✓ PCA 투영 및 역 투영
    - ❑ mean, eVects = cv2.PCACompute(X, mean = None)는 X의 평균 벡터 mean, 공분산 행렬의 고유벡터 eVects를 계산
    - ❑ Y =cv2.PCAProject(X, mean, eVects)는 cv2. PCAProject() 함수는 고유 벡터 eVects에 의해 PCA 투영
    - ❑ 데이터를 고유 벡터를 축으로 한 좌표로 변환
    - ❑ 아래 수식에 A는 직교 행렬로 eVects이고 m은 평균 벡터이며 x, y, m은 2 X 1 열벡터
    - ❑ x는 X의 각 행에 저장된 좌표를 열벡터 변환
    - ❑ Y는 수식의 열벡터 y를 각 행에 저장한 PCA 투영 결과

$$y = A(x - m) \quad \#PCA\ projection$$

$$A^{-1}y = A^{-1}A(x - m)$$

$$A^{T}y = (x - m) \quad \#\text{직교행렬}, A^{-1} = A^{T}$$

$$x = A^{T}y + m \quad \#PCA\ backprojection$$

통계 관련 함수

- ❖ 수학 및 통계 함수
  - ✓ PCA 투영 및 역 투영
    - ❑ X2 = cv2.PCABackProject(Y, mean, eVects)
    - ❑ Y를 PCA 역투영하면 원본 X를 복구할 수 있으며 X와 X2는 오차 범위 내에서 같은 값을 갖는 np.allclose(X, X2)는True

통계 관련 함수

- ❖ 행렬 연산
  - ✓ 행렬의 곱

    cv2.gemm(src1, src2, alpha, src3, beta[, dst[, flags]]) -> dst

    - ❑ 수식: dst = alpha • $srci1^T$ • src2 + beta • $src3^T$
    - ❑ 파라미터
      - src1, src2: 행렬 곱을 위한 두 입력 행렬(np.float32/np.float64형 2채널까지 가능)
      - alpha: 행렬 곱에 대한 가중치
      - src3: 행렬 곱에 더해지는 델타 행렬
      - beta: src3 행렬에 곱해지는 가중치
      - dst: 출력 행렬
      - flags: 옵션을 조합하여 입력 행렬들을 전치

        cv2.GEMM_1_T -> 1, src1을 전치

        cv2.GEMM_2_T -> 2, src2을 전치

        cv2.GEMM_3_T -> 4, src3을 전치

통계 관련 함수

- ❖ 행렬 연산
  - ✓ 행렬 변환: 행렬에 투영 변환을 수행

    cv2.perspectiveTransform(src, m[, dst]) -> dst

    - ❑ 파라미터
      - ○ src: 입력 행렬(2채널 이나 3채널 부동소수점 배열)
      - ○ dst: 출력 행렬
      - ○ m: 투영할 3 X 3, 4 X 4 배열

통계 관련 함수

- ❖ 행렬 연산
  - ✓ 역 행렬

    cv2.invert(src[, dst[, flags]) -> retval, dst

    - ❑ 파라미터
      - ○ src: 입력 행렬
      - ○ dst: 출력 행렬
      - ○ flags: 역행렬 계산 방법

        cv2.DECOMP_LU(0) -> 가우시안 소거법 이용

        cv2.DECOMP_SVD(1) -> 특이값 분해 이용

        cv2.DECOMP_CHOLESKY(2) -> CHOLESKY 분해 이용

통계 관련 함수

- ❖ 행렬 연산
  - ✓ 연립 방정식 풀이

    cv2.solve(src1, src2[, dst[, flags]) -> retval, dst

    - ❑ 파라미터
      - ○ src1: 연립 방정식의 계수 행렬
      - ○ src2: 연립 방정식의 상수 행렬
      - ○ dst: 출력 행렬
      - ○ flags: 역행렬 계산 방법

        cv2.DECOMP_LU(0) -> 가우시안 소거법 이용

        cv2.DECOMP_SVD(1) -> 특이값 분해 이용

        cv2.DECOMP_CHOLESKY(2) -> CHOLESKY 분해 이용