|  |  |
| --- | --- |
| 교육 제목 | Binary tresholding |
| 교육 일시 | 2021. 11. 16 |
| 교육 장소 | 영우글로벌러닝 |
| **교육 내용** | |
| 오전 | import numpy as np  import sys  import cv2  src=cv2.imread('fig/blue\_eyes.png', cv2.IMREAD\_GRAYSCALE)  kernel\_3=np.ones((3,3), np.float32)/9  kernel\_5=np.ones((5,5), np.float32)/25  src\_mean\_filter3=cv2.filter2D(src, -1, kernel\_3) #, (-1, -1), 0)  src\_mean\_filter5=cv2.filter2D(src, -1, kernel\_5)  cv2.imshow('src', src)  cv2.imshow('src\_mean\_filter3', src\_mean\_filter3)  cv2.imshow('src\_mean\_filter5', src\_mean\_filter5)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/blue\_eyes.png', cv2.IMREAD\_GRAYSCALE)  dst\_gaussian1=cv2.GaussianBlur(src, (0,0), 1)  dst\_gaussian2=cv2.GaussianBlur(src, (0,0), 2)  dst\_mean=cv2.blur(src, (7,7))  cv2.imshow('src', src)  cv2.imshow('dst\_mean', dst\_mean)  cv2.imshow('dst\_gaussian1', dst\_gaussian1)  cv2.imshow('dst\_gaussian2', dst\_gaussian2)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/blue\_eyes.png', cv2.IMREAD\_GRAYSCALE)  src\_gblur1=cv2.GaussianBlur(src, (0, 0), 1)  src\_gblur2=cv2.GaussianBlur(src, (0, 0), 2)  dst\_sharp1=cv2.addWeighted(src, 2, src\_gblur1, -1, 0)  dst\_sharp2=cv2.addWeighted(src, 2, src\_gblur2, -1, 0)  cv2.imshow('src', src)  cv2.imshow('src\_gblur1', src\_gblur1)  cv2.imshow('src\_gblur2', src\_gblur2)  cv2.imshow('dst\_sharp1', dst\_sharp1)  cv2.imshow('dst\_sharp2', dst\_sharp2)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/blue\_eyes.png', cv2.IMREAD\_GRAYSCALE)  print(src.shape)  salt\_pepper\_1=np.random.choice((0, 255), src.shape, p=(0.95, 0.05)).astype(np.int32)  salt\_pepper\_2=np.random.choice((0, 255), src.shape, p=(0.95, 0.05)).astype(np.int32)  src\_noise=src+salt\_pepper\_1-salt\_pepper\_2  src\_noise=np.clip(src\_noise, 0, 255).astype(np.uint8)  cv2.imshow('src', src)  # cv2.imshow('salt\_pepper\_1', salt\_pepper\_1)  cv2.imshow('src\_noise', src\_noise)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/blue\_eyes.png', cv2.IMREAD\_GRAYSCALE)  dst\_gaussian=cv2.GaussianBlur(src, (0, 0), 1)  dst\_bilateral=cv2.bilateralFilter(src, -1, 10, 3)  cv2.imshow('src', src)  cv2.imshow('dst\_gaussian', dst\_gaussian)  cv2.imshow('dst\_bilateral', dst\_bilateral)  cv2.waitKey()  cv2.destroyAllWindows()  ----------------------------------------------------------------------------------------  import numpy as np  import sys  import cv2  src=cv2.imread('fig/puppy.bmp')  # M=np.array([[1, 0, -200],  # [0, 1, -100]], np.float32)  M=np.array([[2, 0, 0],  [0, 2, 0]], np.float32)  # warpAffine(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]]) -> dst  # src: 입력영상  # M: affine transform matrix (size: 2 x 3)  # dsize: 출력영상 크기, (0, 0) = 입력영상크기로 출력  # borderValue: 값이 없는 영역을 채우는 값, default = 0  dst=cv2.warpAffine(src, M, (0, 0))  cv2.imshow('src', src)  cv2.imshow('dst', dst)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/rose.jpg')  print(src.shape)  # resize(src, dsize[, dst[, fx[, fy[, interpolation]]]]) -> dst  # src: 입력영상  # dsize: 출력영상의 크기  # fx, fy: 0, 0  # interpolation  dst\_1=cv2.resize(src, (2\*src.shape[0], 2\*src.shape[1]), interpolation=cv2.INTER\_NEAREST)  dst\_2=cv2.resize(src, (2\*src.shape[0], 2\*src.shape[1]), interpolation=cv2.INTER\_LINEAR)  dst\_3=cv2.resize(src, (2\*src.shape[0], 2\*src.shape[1]), interpolation=cv2.INTER\_CUBIC)  dst\_4=cv2.resize(src, (2\*src.shape[0], 2\*src.shape[1]), interpolation=cv2.INTER\_LANCZOS4)  cv2.imshow('src', src)  cv2.imshow('dst\_1', dst\_1)  cv2.imshow('dst\_2', dst\_2)  cv2.imshow('dst\_3', dst\_3)  cv2.imshow('dst\_4', dst\_4)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/puppy.bmp')  # warpAffine(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]]) -> dst  # src: 입력영상  # M: affine transform matrix (size: 2 x 3)  # dsize: 출력영상 크기, (0, 0) = 입력영상크기로 출력  # borderValue: 값이 없는 영역을 채우는 값, default = 0  rad=20\*np.pi/180  rad\_30=30\*np.pi/180  M=np.array([[np.cos(rad), np.sin(rad), 0],  [-np.sin(rad), np.cos(rad), 0]], np.float32)  M1=np.array([[np.cos(rad\_30), np.sin(rad\_30), 0],  [-np.sin(rad\_30), np.cos(rad\_30), 0]], np.float32)  dst=cv2.warpAffine(src, M, (0, 0))  dst2=cv2.warpAffine(src, M1, (0, 0))  # getRotationMatrix2D(center, angle, scale) -> retval  # center: 영상의 center  # angle: 회전각도  # scale: 확대율  center=(src.shape[1]/2, src.shape[0]/2)  print(center)  affine=cv2.getRotationMatrix2D(center, 20, 1)  dst\_rot=cv2.warpAffine(src, affine, (0, 0))  cv2.imshow('src', src)  cv2.imshow('dst', dst)  cv2.imshow('dst\_rot', dst\_rot)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/rose.jpg')  rad=20\*np.pi/180  M=np.array([[np.cos(rad), np.sin(rad), 0],  [-np.sin(rad), np.cos(rad), 0]], np.float32)  dst=cv2.warpAffine(src, M, (0, 0))  center=(src.shape[1]/2, src.shape[0]/2)  print(center)  affine=cv2.getRotationMatrix2D(center, 20, 1)  dst\_rot=cv2.warpAffine(src, affine, (0, 0), borderMode=cv2.BORDER\_REPLICATE)  cv2.imshow('src', src)  cv2.imshow('dst', dst)  cv2.imshow('dst\_rot', dst\_rot)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/puppy.bmp')  M=np.array([[1, 0.2, 0],  [0, 1, 0]], np.float32)  h, w=src.shape[:2]  print(h, w)  dst=cv2.warpAffine(src, M, (w+100, h))  cv2.imshow('src', src)  cv2.imshow('dst', dst)  cv2.waitKey()  cv2.destroyAllWindows() |
| 오후 | import numpy as np  import sys  import cv2  src=cv2.imread('fig/lenna.bmp', cv2.IMREAD\_GRAYSCALE)  kernel\_dx=np.array([[-1, 0, 1],  [-2, 0, 2],  [-1, 0, 1]], dtype=np.float32)  print(kernel\_dx)  # cv2.Sobel(src, ddepth, dx, dy, dst, ksize, scale, delt, borderType) -> dst  # src : 입력영상  # ddepth : 출력영상의 데이터 타입 (-1)  # dx : x 방향 미분차수  # dy : x 방향 미분차수  # dst : 출력영상  # ksize : 커널의 크기  # scale : 연산결과에 추가적으로 곱할 값  # delta : 연산결과에 추가적으로 더할 값  # borderType : 가장자리 픽셀확장 방식  # dst\_dx=cv2.filter2D(src, -1, kernel\_dx, delta=128)  dx=cv2.Sobel(src, cv2.CV\_32F, 1,0)#, delta=128)  dy=cv2.Sobel(src, cv2.CV\_32F, 0,1)#, delta=128)  mag=np.clip(cv2.magnitude(dx, dy), 0, 255).astype(np.uint8)  dx=np.clip(dx, 0, 255).astype(np.uint8)  dy=np.clip(dy, 0, 255).astype(np.uint8)  dst=np.zeros(mag.shape[:2], np.uint8)  dst[mag>100]=255  cv2.imshow('src', src)  # cv2.imshow('dx', dx)  # cv2.imshow('dy', dy)  cv2.imshow('mag', mag)  cv2.imshow('dst', dst)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/oo.jpg', cv2.IMREAD\_COLOR)  kernel\_dx=np.array([[-1, 0, 1],  [-2, 0, 2],  [-1, 0, 1]], dtype=np.float32)  dx=cv2.Sobel(src, cv2.CV\_32F, 1,0)  dy=cv2.Sobel(src, cv2.CV\_32F, 0,1)  mag=np.clip(cv2.magnitude(dx, dy), 0, 255).astype(np.uint8)  dst\_sobel=np.zeros(mag.shape, np.uint8)  dst\_sobel[mag>100]=255  dst\_canny=cv2.Canny(src, 100, 150)  cv2.imshow('src', src)  cv2.imshow('dst\_sobel', dst\_sobel)  cv2.imshow('dst\_canny', dst\_canny)  cv2.waitKey()  cv2.destroyAllWindows()  ---------------------------------------------------------------------------------  import numpy as np  import sys  import cv2  src=cv2.imread('fig/cells.png', cv2.IMREAD\_GRAYSCALE)    retval1, dst1=cv2.threshold(src, 100, 255, cv2.THRESH\_BINARY)  retval2, dst2=cv2.threshold(src, 200, 255, cv2.THRESH\_BINARY)  print(retval1, retval2)  cv2.imshow('src', src)  cv2.imshow('dst1', dst1)  cv2.imshow('dst2', dst2)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/cells.png', cv2.IMREAD\_GRAYSCALE)  def call\_trackbar(pos):  retval, dst=cv2.threshold(src, pos, 255, cv2.THRESH\_BINARY\_INV)  cv2.imshow('dst', dst)  cv2.imshow('src', src)  cv2.namedWindow('dst')  cv2.createTrackbar('level', 'dst', 128, 255, call\_trackbar)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/rice.png', cv2.IMREAD\_GRAYSCALE)  thres, dst=cv2.threshold(src, 0, 255, cv2.THRESH\_OTSU)  print(thres)  cv2.imshow('src', src)  cv2.imshow('dst', dst)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/rice.png', cv2.IMREAD\_GRAYSCALE)  thres, dst=cv2.threshold(src, 0, 255, cv2.THRESH\_OTSU)  dst2=np.zeros(src.shape, np.uint8)  bw=src.shape[1]//4  bh=src.shape[0]//4  for y in range(4):  for x in range(4):  src\_=src[y\*bh:(y+1)\*bh, x\*bw:(x+1)\*bw]  dst\_=dst2[y\*bh:(y+1)\*bh, x\*bw:(x+1)\*bw]  cv2.threshold(src\_, 0, 255, cv2.THRESH\_OTSU, dst\_)  cv2.imshow('src', src)  cv2.imshow('dst', dst)  cv2.imshow('dst2', dst2)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/sudoku.jpg', cv2.IMREAD\_GRAYSCALE)  # cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C, dst) -> dst  # src: 입력영상  # maxValue: 임계값 최대치, 255  # adaptiveMethod: 지역이진화 전 블러링함수, 예 cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C/cv2.ADAPTIVE\_THRESH\_mean\_C  # thresholdType:cv2.THRESH\_BINARY 또는 cv2.THRESH\_BINARY\_INV  # blocksize: 3이상의 홀수  # C: 블록 내 평균괎  dst=cv2.adaptiveThreshold(src, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY,11, 5)  cv2.imshow('src', src)  cv2.imshow('dst', dst)  # cv2.imshow('dst2', dst2)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/circuit.bmp', cv2.IMREAD\_GRAYSCALE)  kernel=cv2.getStructuringElement(cv2.MORPH\_RECT, (3, 5))  dst1=cv2.erode(src, kernel)  dst2=cv2.dilate(src, kernel)  dst3=cv2.erode(dst2, kernel)  cv2.imshow('src', src)  cv2.imshow('dst1', dst1)  cv2.imshow('dst2', dst2)  cv2.imshow('dst3', dst3)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/rice.png', cv2.IMREAD\_GRAYSCALE)  dst=np.zeros(src.shape, np.uint8)  bw=src.shape[1]//4  bh=src.shape[0]//4  for y in range(4):  for x in range(4):  src\_=src[y\*bh:(y+1)\*bh, x\*bw:(x+1)\*bw]  dst\_=dst[y\*bh:(y+1)\*bh, x\*bw:(x+1)\*bw]  cv2.threshold(src\_, 0, 255, cv2.THRESH\_OTSU, dst\_)  # cv2.morphologyEx(src, op, kernel, dst, anchor, iterations, borderType, borderValue) -> dst  # src: 입력영상  # op: morphology 연산 플래스  # cv2.MORPH\_ERODE  # cv2.MORPH\_DILATE  # cv2.MORPH\_OPEN  # cv2.MORPH\_CLOSE  # cv2.MORPH\_GRADIENT (팽창 - 침식)  kernel=cv2.getStructuringElement(cv2.MORPH\_RECT, (3, 3))  dst2=cv2.morphologyEx(dst, cv2.MORPH\_OPEN, kernel, None)  dst3=cv2.morphologyEx(dst, cv2.MORPH\_ERODE, kernel, None)  cv2.imshow('src', src)  cv2.imshow('dst', dst)  cv2.imshow('dst2', dst2)  cv2.imshow('dst3', dst3)  cv2.waitKey()  cv2.destroyAllWindows()  src=cv2.imread('fig/keyboard.jpg', cv2.IMREAD\_GRAYSCALE)  ret, dst=cv2.threshold(src, 0, 255, cv2.THRESH\_OTSU)  kernel=cv2.getStructuringElement(cv2.MORPH\_RECT, (3, 3))  dst\_morph=cv2.morphologyEx(dst, cv2.MORPH\_OPEN, kernel)  # connectedComponents(image[, labels[, connectivity[, ltype]]]) -> retval, labels  cnt, labels, stats, centroids=cv2.connectedComponentsWithStats(dst\_morph)  dst=cv2.cvtColor(src, cv2.COLOR\_GRAY2BGR)  for i in range(1, cnt):  x, y, w, h, area=stats[i]    if area>1000 or area<100:  continue    cv2.rectangle(dst, (x, y, w, h), (0, 0, 255), 2)    cv2.imshow('src', src)  cv2.imshow('dst', dst)  # cv2.imshow('dst\_morph', dst\_morph)  # cv2.imshow('dst3', dst3)  cv2.waitKey()  cv2.destroyAllWindows() |