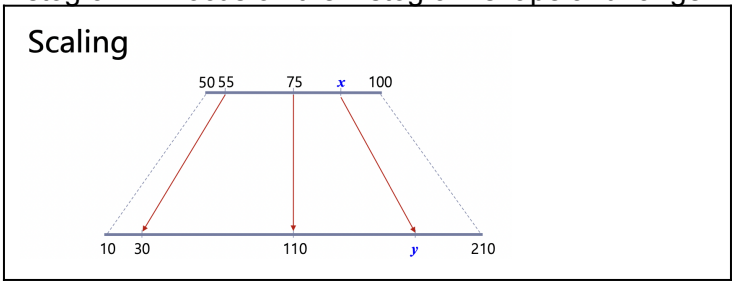
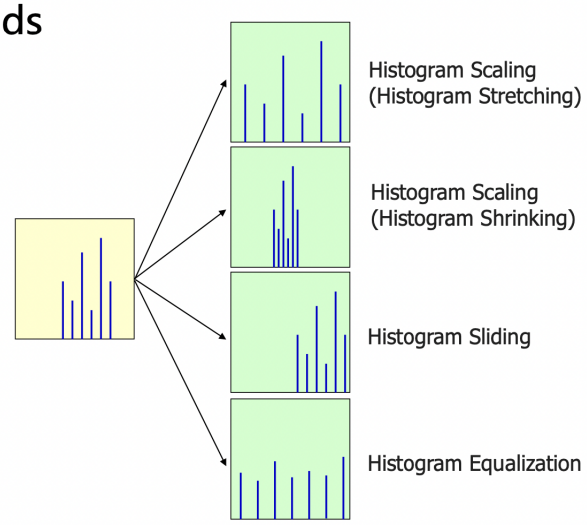
교육 제목	데이터 기반 인공지능 시스템 엔지니어 양성 과정
교육 일시	2021년 11월03/05일
교육 장소	YGL C-6 학과장 & 자택(디스코드 이용한 온라인)
교육 내용	
오전	VISION 1. 2차원 신호의 디지털화 과정 : Sampling -> Quantization -> Coding ○ Sampling : 이미지의 pixel화 ○ Quantization : pixel의 level화 ○ Coding : 처리된 이미지의 압축 2. 디지털 영상의 구조 및 유형 ○ 디지털 영상의 표현 방법 : 영상좌표 (x,y), 단, 0이 좌측 상단에 있음 (c.f. 행렬 위치 (r, c) -> 영상좌표와 반대이다.) -> bitmap = I (x,y) 참고! ■ 1은 흰색, 0은 검은색 ■ Sampling : dpi(분해능, resolution) 증가 -> 이미지/처리량 증가, Quantization (양자화) 증가 -> 이미지/처리량 증가 ■ 2lvl = binary = Mask 영상 ■ 영상처리의 목적은 눈에 보기 좋게 만드는 것이다. (c.f. 의료영상은 많은 정보를 담는 것이 목적이다.) ■ 영상처리(Image Processing) --> Computer Vision ■ Vectorized image (pdf 파일, 확대해도 이미지가 안깨짐) (.svg, .eps 형태의 파일) ■ catesian 좌표계 (일반), polar 좌표계 (의료영상) ■ 영상 좌표와 행렬 좌표가 다르므로 컴퓨팅 처리를 위해 numpy arrary로 변경할 때 주의해야 한다. ■ jpg : true color(rgb), indexed color(x), lossy(o) ■ bmp : indexed, true(x), gray scale (o) ■ gif : indexed, true (x), gray scale (x) ■ png : lossy(x) ○ 디지털 영상의 유형(mode) - binary : 1 bit / pixel (눈속임 방법 : dithering, Halftoning) - grayscale : 8 bit /pixel - color : 24 bit / pixel (=16,777,216 colors) // RGB - multi-spectral : 여러 이미지가 중첩된 이미지 3. 영상 처리 및 컴퓨터비전 개요 ○ computer imaging where the application does not involve a human being in the visual loop ■ the images are examined and acted upon by a computer ■ the final application requires a computer to use the visual information directly ○ Fields of computer Vision

        - factory automation system : quality control, inspection
        - automatic identification (biometrics system) : DNA analysis, fingerprints identification, retinal identification
        - medical diagnostic system
        - intelligent transport system
    - Pattern recognition : input object(pattern)를 주어진 algorithm에 의해 category 나 class 로 classification 하는 과정
        - 절차 : image enhancement -> image segmentation -> feature extraction -> pattern classification
    - image processing 개요
        - 인간이 사용하기 편하도록 컴퓨터를 사용하여 기존 영상을 개선하거나 수정하는 학문 분야(images are to be examined and acted upon by people)
        - fields of image processing
            - image restoration (복원)
            - image enhancement (개선)
            - image compression (압축)

4. Arithmetic Operation & Grayscale Transformation
    - Point Operation 개요
        - Each pixel value is replaced with a new value obtained from the old one : I(x,y) →f(x,y) → O(x,y)
    - Arithmetic Operation : O(x,y)_ = k * I(x,y) + I
      (클리핑(clipping)처리 : if (O(x,y) > 255), O(x,y) = 255;
      if (O(x,y) < 0), O(x,y) = 0
        - image contrast : a measure of the distribution and range of the gray levels
        - image brightness: the overall average or mean pixel value in the image
        - averaging
        - thresholding
    - Grayscale Transformations : O(x,y) = M [I(x,y)]
        - improving image contrast and brightness by using mapping function
        - grayscale compression
        - grayscale stretching
        - gray-level negative
    - Processing for Color images
5. Histogram Modification
    - Histogram
        - a simple datum that gives the number of pixels that a given value in an image
    - Histogram Modifications
        - improving image contrast and brightness based on histogram → focus on the histogram shape and range



fields



Scaling

$$\mathbf{O}(x,y) = \left[\frac{S_{max} - S_{min}}{I_{max} - I_{min}}\right][\mathbf{I}(x,y) - I_{min}] + S_{min}$$

I_{max} : largest gray-level value in the image $\mathbf{I}(x,y)$
I_{min} : smallest gray-level value in $\mathbf{I}(x,y)$
S_{max} : maximum gray-level values possible
S_{min} : minimum gray-level values possible

Sliding

$$\mathbf{O}(x,y) = \mathbf{I}(x,y) + offset$$

offset : amount to slide the histogram

오후

OpenCV-Python 기초사용법

영상속성

[1]:
```
import sys
import cv2
import numpy as np
```

[2]:
```
# cv2.imread(filename[, flags]) -> retval

# retval
    # numpy.nbarray: retval.ndim/shape/size/dtype
    # dtype: uint8
    # shape: gray영상의 경우 (h,w) 또는 color (h,w, 3)
    # gray영상 : cv2.CV_8UC1 -> numpy.uint8
    # color영상 : cv2.CV_8UC3 -> numpy.uint8
```

[3]:
```
# 영상 불러오기
img1 = cv2.imread('fig/puppy.bmp', cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread('fig/puppy_1280_853.jpg', cv2.IMREAD_COLOR)

if img1 is None or img2 is None:
    print('Image load failed!')
    sys.exit()


# 영상의 속성 참조
print('type(img1):', type(img1))
print('img1.shape:', img1.shape)
print('img2.shape:', img2.shape)
print('img1.dtype:', img1.dtype)
print('img2.dtype:', img2.dtype)

print('img1.shape length:', len(img1.shape))
print('img2.shape length:', len(img2.shape))
```

•••

영상의 크기 참조

[4]:
```
h, w = img1.shape
print('img1 size: {} x {}'.format(w, h))

h, w = img2.shape[:2]
print('img2 size: {} x {}'.format(w, h))
```

•••

영상의 크기 참조

[4]:
```
h, w = img1.shape
print('img1 size: {} x {}'.format(w, h))

h, w = img2.shape[:2]
print('img2 size: {} x {}'.format(w, h))
```

•••

영상의 픽셀값 참조

[7]:
```
x = 230
y = 320

p1 =img1[y,x]
print(p1)

p2 = img2[y, x]
print(p2)

### 픽셀값 바꾸기
img1[10:200, 10:200] = 0
img2[10:200, 10:200] = (0, 0, 255)

cv2.imshow('image', img1)
cv2.imshow('image2',img2)

cv2.waitKey()
cv2.destroyAllWindows()
```

영상생성

[9]:
```
import numpy as np
import cv2

# 새 영상 생성하기
# img1 = np.empty((240, 320), dtype=np.uint8)       # grayscale image
img1 = np.random.randint(0, 255, (240, 320), dtype = np.uint8) # gray random scale
img2 = np.zeros((240, 320, 3), dtype=np.uint8)    # color image
img3 = np.ones((240, 320), dtype=np.uint8) * 255  # dark gray
img4 = np.full((240, 320, 3), (0, 255, 255), dtype=np.uint8)  # yellow

cv2.imshow('img1', img1)
cv2.imshow('img2', img2)
cv2.imshow('img3', img3)
cv2.imshow('img4', img4)

cv2.waitKey()
cv2.destroyAllWindows()
```

새영상 생성

[30]:
```python
# 영상 복사
img1 = cv2.imread('fig/puppy.bmp', cv2.IMREAD_COLOR)
# img1 = cv2.imread('HappyFish.jpg')

if img1 is None:
    print("image load failed")
    sys.exit()

img2 = img1
# img2 = img1[150:250, 200:500]

img3 = img1.copy()

# img2[:] = (0, 0, 255)

img1[200:300,240:400] = (0, 255, 255)

print(img1.shape)
print(img1.dtype)

cv2.imshow('img1', img1)
cv2.imshow('img2', img2)
cv2.imshow('img3', img3)

cv2.waitKey()
cv2.destroyAllWindows()
```

부분 영상 추출

[28]:
```python
img1 = cv2.imread('fig/puppy.bmp')

img2 = img1[200:400, 300:500]  # numpy.ndarray의 슬라이싱
img3 = img1[200:400, 300:500].copy()

# img1.fill(255)
cv2.circle(img2, (100, 100), 50, (0, 0, 255), 3)

cv2.imshow('img1', img1)
cv2.imshow('img2', img2)
cv2.imshow('img3', img3)


cv2.waitKey()
cv2.destroyAllWindows()
```

마스크 연산과 ROI

[33]:
```python
# 마스크 영상을 이용한 영상 합성
# cv2.copyTo(src, mask, dst = None) -> dst

src = cv2.imread('fig/airplane.bmp', cv2.IMREAD_COLOR)
mask = cv2.imread('fig/mask_plane.bmp', cv2.IMREAD_GRAYSCALE)
dst = cv2.imread('fig/field.bmp', cv2.IMREAD_COLOR)

if src is None or mask is None or dst is None:
    print('Image read failed!')
    sys.exit()


# 영상의 포맷과 형식이 같아야 함
cv2.copyTo(src, mask, dst)
# dst = cv2.copyTo(src, mask)

# Using numpy
# dst[mask > 0] = src[mask > 0]

cv2.imshow('src', src)
cv2.imshow('dst', dst)
cv2.imshow('mask', mask)

cv2.waitKey()
cv2.destroyAllWindows()
```

알파 채널을 마스크 영상으로 이용

[50]:
```python
src = cv2.imread('fig/puppy.bmp', cv2.IMREAD_COLOR)
sunglass = cv2.imread('fig/imgbin_sunglasses_1.png', cv2.IMREAD_UNCHANGED)

sunglass = cv2.resize(sunglass, (300, 150))

if src is None or sunglass is None:
    print('Image read failed!')
    sys.exit()

# 0(완전 투명) ~ 255(완전 불투명)
mask = sunglass[:, :, -1]    # mask는 알파 채널로 만든 마스크 영상
glass = sunglass[:, :, 0:3]  # glass는 b, g, r 3채널로 구성된 컬러 영상

h, w = mask.shape[:2]
crop = src[120:120+h, 220:220+w]  # glass mask와 같은 크기의 부분 영상 추출

cv2.copyTo(glass, mask, crop)
# crop[mask > 0] = (0, 0, 255)

cv2.imshow('src', src)
cv2.imshow('glass', glass)
cv2.imshow('mask', mask)
# cv2.imshow('crop', crop)

# cv2.imwrite('puppy_sunglass.bmp', src)

cv2.waitKey()
cv2.destroyAllWindows()
```