|  |  |
| --- | --- |
| 교육 제목 | **화소의 처리 방법 OpenCV 활용** |
| 교육 일시 | 2021년 12월 15일 수요일 |
| 교육 장소 | 영우글로벌러닝 2층 |
| **교육 내용** | |
| 오전 | 1. 화소의 영역 처리    1. 출력 영상의 새로운 화소 값을 결정하려고 해당 입력 화소뿐만 아니라 그 주변의 화소도 함께 고려하는 공간 영역 연산    2. 회선 기법(처리) 또는 컨벌루션 처리라고도 함. 2. 회선 기법    1. 주변 화소의 조합을 결정하여 새로운 화소를 출력해 줌.    2. 원시 화소에 이웃한 각 화소에 가중치를 곱한 합을 출력 화소로 생성 3. 화소의 영역 기반 처리    1. 엠보싱 효과, 블러링, 샤프닝, 경계선 검출, 잡음 제거 등이 있음 4. 엠보싱 효과    1. 입력 영상을 양각 형태로 나타냄=영상의 특정 부분이 볼록해 보이도록 만듦. 5. 블러링    1. 영상의 세밀한 부분을 제거하여 영상을 흐리게 하거나 부드럽게 나타내는 기술. 고 주파 성분을 제거하는 기술(영상의 세밀한 부분은 고주파 성분). 6. 경계선 검출    1. 디지털 영상에 있는 경계선(Edge)을 찾아내는 기법 7. 화소의 영역 처리를 수행하는 디지털 영상처리 시스템은 선형 시불변 시스템을 만족시킴 8. 디지털 영상처리의 결과는 컨벌루션 또는 회선 처리로 얻을 수 있음 9. 회선 기법으로 생성되는 새로운 화소 값 10. 이웃 화소 값과 이에 대응하는 회선 마스크의 가중치를 곱한 뒤 곱한 값을 더해서 얻음. 여기서 가중치는 작은 행렬인 회선 마스크 또는 회선 커널로 구성됨. 11. 디지털 영상에서 화소의 영역 처리를 수행하는 회선 기법 12. 가중치를 포함한 회선 마스크가 이동하면서 수행 13. 경계 부분 처리 14. 회선 마스크에 대응할 요소가 없는 영상의 화소를 처리하는 방법 15. 영상에서의 주파수 개념은 화소 값의 변화율을 나타냄. 16. 블러링 처리나 스무딩 처리를 하려고 영상처리에서 사용하는 대표적인 저역 통 과 필터로 가우시안 필터가 있음. |
| 오후 | 1. Normalization(정규화)    1. 영상에서 특정 부분에 몰려 있는 값을 전체 영역으로 골고루 분포하게 하도록 만드는 방법을 Normalization(정규화) 라고 한다. 2. Equalization(평탄화)    1. 이미지의 히스토그램이 전체 영역에 골고루 분포가 되어 있을 때 좋은 이미지라고 할 수 있다. 아래 히스토그램을 보면 왼쪽 그래프에서 가운데에 분포가 집중되어 있는데, 이를 오른쪽 그래프처럼 만들어주는 작업을 Histogram Equalization이라고 한다. 노멀라이즈의 방법에서 한 곳에 집중된 영역으로부터 멀리 떨어진 값이 있을 경우에는 효과가 없는데, 그런 경우에 사용할 수 있다.    2. 이퀄라이즈는 각각의 값이 전체 분포에 차지하는 비중에 따라 분포를 재분배하므로 명암 대비(contrast)를 개선하는데 효과적이다. 3. 영상의 속성과 픽셀 값 참조    1. OpenCV는 영상 데이터를 numpy.ndarray로 표현합니다.       1. 영상의 속성을 참조하는 명령어는 다음과 같습니다.       2. .ndim : 차원 수, len(img.shape)와 같음       3. .shape : 각 차원의 크기, 흑백(h,w), 컬러(h,w,3), png(h,w,4)       4. .size : 전체 원소 개수       5. .dtype : 원소의 데이터 타입. 영상 데이터는 unit8 4. 영상의 픽셀 값 참조    1. img1 = cv2.imread('cat.bmp', cv2.IMREAD\_GRAYSCALE) # 그레이스케일    2. img2 = cv2.imread('cat.bmp', cv2.IMREAD\_COLOR) # 컬러    3. # for문을 이용한 영상의 픽셀 값 수정    4. for y in range(h):    5. for x in range(w):    6. # y, x 순서인 이유 : 영상 행렬은 높이, 길이로 저장되므로    7. img1[y, x] = 255 # BGR 모두 255 => 하양색    8. img2[y, x] = (0, 0, 255) # x,y 위치의 픽셀을 (0, 0, 255) 빨강색으로 만듬    9. # 슬라이싱을 이용한 영상의 픽셀 값 수정    10. img1[:,:] = 255 # 모든 픽셀을 흰색    11. img2[:,:] = (0, 0, 255) # 모든 픽샐을 빨강색    12. # 영상 출력    13. cv2.imshow('img1', img1)    14. cv2.imshow('img2', img2)    15. cv2.waitKey()    16. # 윈도우 창 닫기    17. cv2.destroyAllWindows() |