chap 11 이진화와 모폴로지

영상의 이진화

이진화

- 영상의 각 픽셀을 두개의 분류로 나누는 작업 -- 입력 영상을 주요 객체 영역과 배경 영역으로 구분. -- 영상에서 중요도가 높은 관심영역, 비관심영역으로 구분
- 영상의 이진화는 0 or 255
- 이진화 적용 영상은 보통 흰색 or 검은색
- 영상의 이진화는 기본적으로 영상의 각 픽셀 값을 이용
- 그레이스케일 영상에 대해 이진화를 수행하려면 영상의 픽셀값이 특정 값보다 크면 255로 설정하고, 작으면 0으로 설정
- 이때 각 픽셀과의 크기 비교 대상이 되는 값을 임계값이라 함. 임계값은 0~255 사이의 정수를 지정
- 임계값은 사용자가 임의 지정 or 영상의 특성을 분석하여 자동으로 결정할 수 있음 -> 임계값은 목적에 따라 적절하게 결정되어야 함.

threshold()함수

- 임계값을 이용한 다양한 연산을 지원하는 함수.
- 열거형 상수를 통해 함수의 동작이 결정됨. -- THRESH_BINARY - 기본동작 -- THRESH_BINARY_INV - 반전을 시킴 -- THRESH_TRUNC - 크면 넣고, 아니면 원본값 -- THRESH_TOZERO - 크면 원본, 아니면 0 -- THRESH_TOZERO_INV - 크면 0, 아니면 원본값 -- THRESH_OTSU - 오츠 알고리즘으로 자동 임계값 설정 - 앞선 5개와 | 연산자로 조합해서 사용 -- THRESH_TRIANGLE - 삼각 알고리즘으로 자동 임계값 설정 - 앞선 5개와 | 연산자로 조합해서 사용

이진화 실습 예제

```
def on threshold(pos):
    _, dst = cv.threshold(src, pos, 255, cv.THRESH_BINARY)
    cv.imshow('dst',dst)

src = cv.imread(IMGPATH, cv.IMREAD_GRAYSCALE)

cv.nameWindow('dst')
cv.createTrackbar('threshold', 'dst', 0, 255, on_threshold)
cv.setTrackbarPos('threshold', 'dst', 128)
```

적응형 이진화

- 영상의 모든 픽셀에 같은 임계값을 적용하여 이진화를 수행하는 방식을 전역 이진화 라고 함.
- 그러나 영상 특성상 전역 이진화를 적용하기 어려운 경우가 있음
- 불균일한 조명 성분을 가진 영상에 대해서는 하나의 임계값으로 객체와 배경을 제대로 구분하기 어려움.
- 그래서 각 픽셀마다 서로 다른 임계값을 사용하는 적응형 이진화 기법을 사용하는 것이 효과적
- 적응형 이진화는 영상의 모든 픽셀에서 정해진 크기와 사각형 블록 영역을 설정하고 블록 영역 내부의 픽셀 값 분포로 고유 임계값을 결정하여 이진화

adaptiveThreshold() 함수 -> 적응형 이진화 함수와 예제

```
def on_trackbar(pos):
    bsize = pos
    if bisze % 2 == 0: bsize = bsize - 1
    if bisze < 3: bisze = 3

    dst = cv.adaptiveThreshold(src, 255, cv.ADAPTIVE_THRESH_GAUSSIAN_C,
cv.THRESH_BINARY, bisze, 5)

    cv.imshow('dst',dst)

src = imread()

cv.imshow('src',src)

cv.nameWindow('dst')

cv.createTrackbar('bloack size', 'dst', 0,200, on_trackbar)
cv.setTrackbarPos('bloack size', 'dst', 11)
```

-> 블럭 사이즈에 따라 임계값이 변하면서 조명이 불균일한 사진에 대해서 이진화가 잘 처리됨.

모폴로지 연산

- 형태 또는 모양에 대한 학문. 영상에서 객체의 형태 및 구조에 대해 분석하고 처리하는 기법.
- 주로 이진영상에서 객체의 모양을 단순화 시키거나 잡음을 제거하는 용도로 사용.
- 구조요소를 정의 -> 구조요소는 마스크처럼 모폴로지 연산의 동작을 결정하는 작은 크기의

구조요소

- 필요에 따라 원하는 구조요소를 선택할 수 있지만 대부분은 3*3 정방향 구조요소를 사용
- 대부분의 경우 중심을 고정점으로 사용.
- openCV에서는 구조 요소 행렬을 간단하게 생성할 수 있도록 getStructuringElement()함수를 제공

침식과 팽창

- 모폴로지 기법의 가장 기본이 되는 연산.

침식 (erosion)

- 침식 연산은 객체영역의 외곽을 골고루 깎아 내는 연산으로 전체적으로 객체 영역은 축소되고, 배경은 확대됨.
- 침식 연산은 구조요소를 영상 전체에 대해 스캔하면서 구조 요소가 객체 영역 내부에 완전히 포함될 경우 고정점 위치를 255로 설정. ->즉, 구조요소(마스크)에 객체의 모든 픽셀이 속해야 고정점을 255로 설정

팽창 (dilation)

- 팽창 연산은 객체 외곽을 확대하는 연산
- 팽창 연산을 수행하면 객체 영역은 확대 되고, 배경 영역은 줄어듦.
- 팽창 연산은 구조요소와 객체 영역이 한 픽셀이라도 만날 경우 고정점 위치 픽셀을 255로 설정. ->즉, 구조요소(마스크)에 객체의 한 픽셀이라도 속하면 고정점을 255로 설정

이진 영상의 침식과 팽창 예제

```
def erode_dilate():
    src = cv.imread(IMGPATH, cv.IMREAD_GRAYSCALE)

    _, src_bin = cv.threshold(src, 0, 255 , cv.THRESH_BINARY |
cv.THRESH_OTSU)

    dst1 = cv.erode(src_bin, None)
    ## 침식연산 함수

    dst2 = cv.dilate(src_bin, None)
    ## 팽창연산 함수
```

이진영상의 열기와 닫기

- 입력 영상에 대해서 침식연산을 수행한 후, 다시 팽창연산을 수행하는것을 열기 연산
- 입력 영상에 대해서 팽창연산을 수행한 후, 다시 침식연산을 수행하는것을 닫기 연산

-> 객체의 영역 크기가 크게 바뀌지 않지만 순서에 따라 다른 효과 발생

열기연산 : 작은 크기의 객체를 효과적으로 제거.

닫기연산 : 객체 내부의 작은 구멍이 매꿔지는 효과.

-> 객체 영역을 확실히 구분하고 싶다면 열기, 이어짐을 확인하고 싶다면 닫기연산을 수행하면 효과적

morphologyEx()함수를 통한 열기, 닫기 예제

```
def open_close():
    src = cv.imread()

    _, src_bin = cv.threshold(src, 0, 255, cv.THRESH_BINARY |
```

```
cv.THRESH_OTSU)

    dst1 = cv.morphologyEx(src_bin, cv.MORPH_OPEN, None)
    # 열기연산
    dst2 = cv.morphologyEx(src_bin, cv.MORPH_CLOSE, None)
    # 닫기 연산
```