ostu’s algorithm

(3.18일 cv수업)

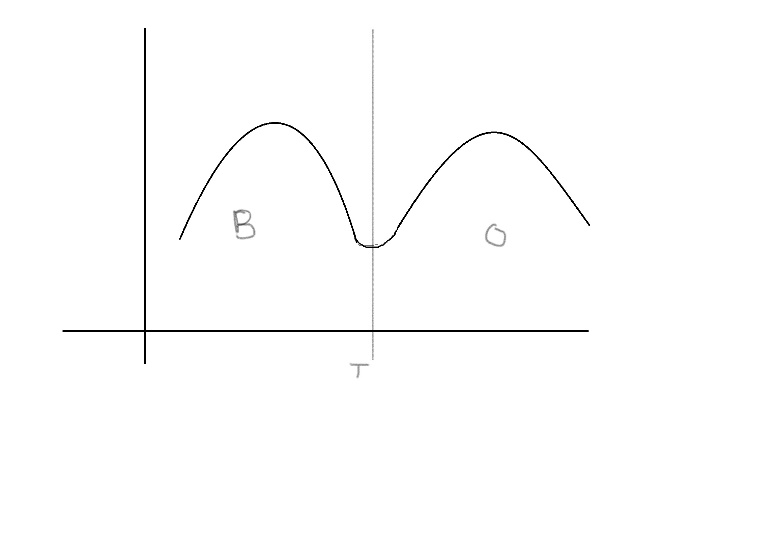
이미지를 인식할 때, 배경과 객체를 분리하는 작업을 한다. 이 작업 중 하나가 이진화 작업이다.

이진화 작업은 객체는 흰 색으로, 배경은 검은 색으로 만들어준다.

이진화 작업 전, 이미지를 grayscale한 뒤, histogram을 생성한다.

(0화소가 몇 개냐, 1화소가 몇 개냐 등…을 수집해서 배열로 만들고, 그를 histogram으로 만든다)

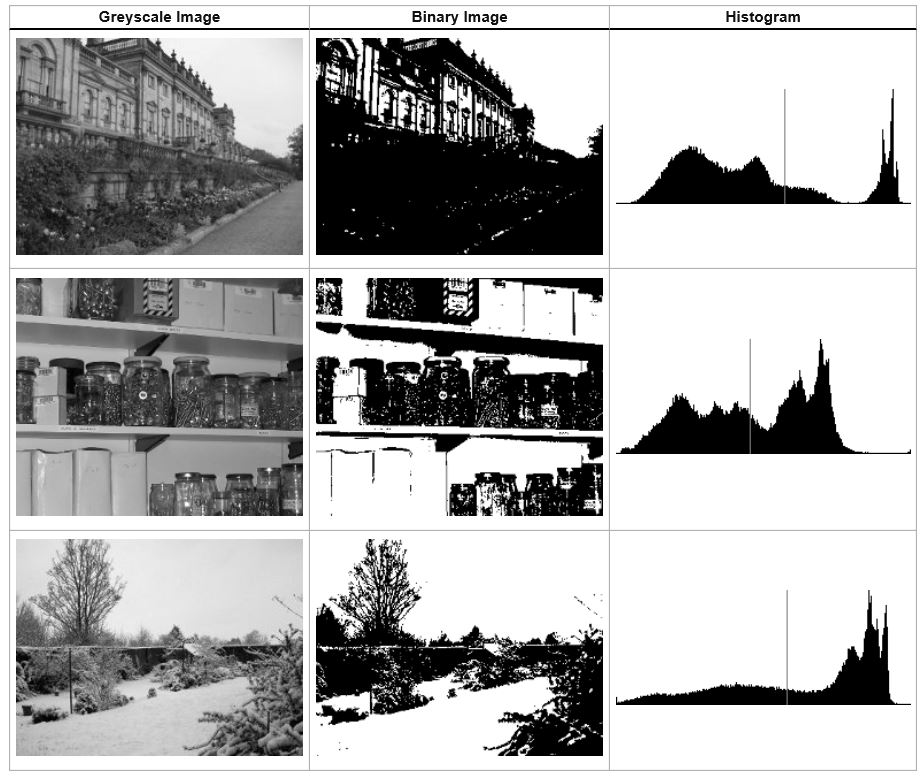
대략



이런 모양이 된다. (배경(B)색이 더 어두우니까 숫자가 낮고, 객체(O)색이 더 밝으므로 숫자가 높다.)

이 사이를 구별해 주는 것이 임계값(T)이고, 각각 프로젝트의 환경에 따라 그래프가 변하므로, 임계값도 바꿔 주어야 한다.

위의 그림은 매우 대략적이고 이상적인 그림이고, 사실은



이런 식으로 히스토그램이 분포된다.

이 임계값을 찾기 위해, hill climbing 방법, 그래프의 평균을 통해 그래프의 표면을 깎아(?)주는 방법, 가중치를 곱해주는 방법 등을 사용하였으나, 결국엔 ostu의 알고리즘을 많은 사람들이 쓰게 되었다. (성능이 ostu것이 가장 좋은가 봅니다)

ostu’s algorithm : (대략적인 아웃라인)

알고리즘을 통해 배경과 객체, 2개의 클래스로 분리한다.

만약 분리를 좋게 했다면, 각 클래스의 그레이값이 비슷할 것이다.(=분산이 작을 것이다)

예시로, 픽셀값? 0~255까지 돌면서 클래스를 분리한다.

(0 | 1~255 , 0~1 | 2~255 등.. 임계값을 반복문으로 하나씩 대입해본다?? 비슷한 개념인듯,,)

분리된 클래스의 분산을 구해서 더한다. (0 | 1~255 : 1234, 0~1 | 2~255 : 2345 …)

각 반복에서 분산이 가장 최소값일 때가 적절한 임계값이다.

내용 참고 출처 : <https://kipl.tistory.com/17> + 수업

이미지 출처 : http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html