

**과목명 | 디지털영상처리**

**실습 | 4**

**담당교수 | 정진우**

**학과 | 컴퓨터공학과**

**학년 | 3**

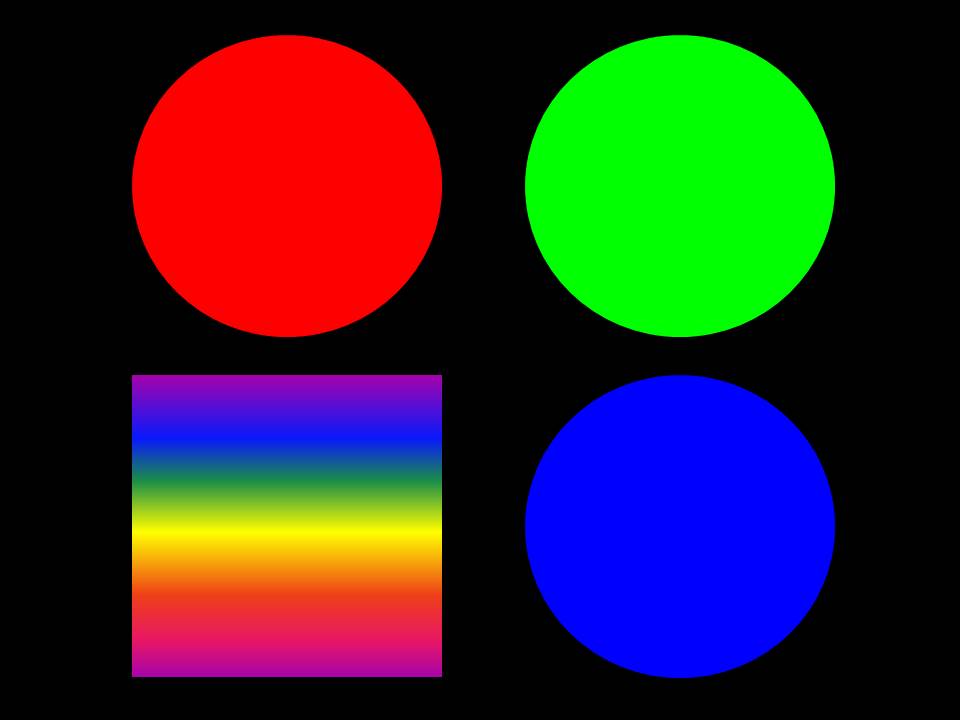
**학번 | 2010112406**

**제출일자 | 2016. 4. 14**

**이름 | 박도령**

**제출일 | 2016. 3. 31**

**실습에 사용한 이미지)**

** **

**mylove.jpg stuff\_color,jpg**

**실습 1)**

♋ **동작 순서 및 원리**

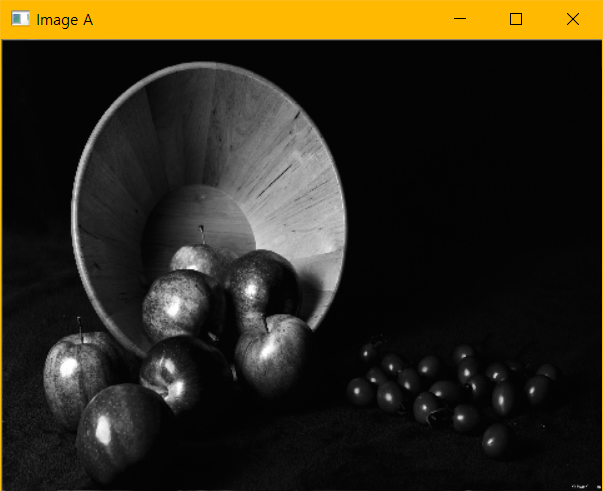
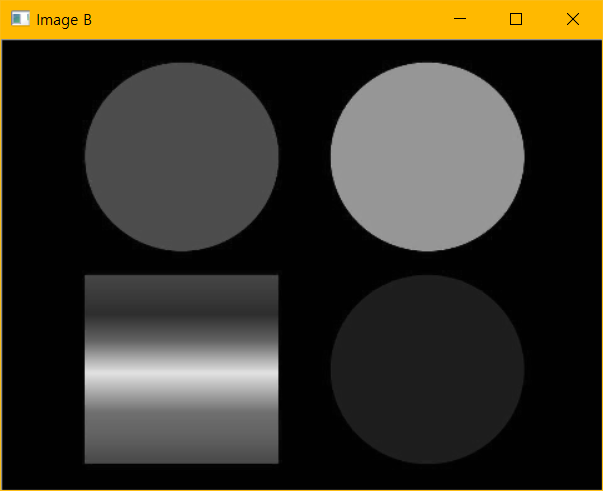
1. **Convert two images to gray scale images   
   (Input : A= any image, b= stuff\_color.jpg file in given materials)**

위의 두 이미지를 로드 한 뒤 Grayscale Image로 변환한다.

이 때 Opencv에서 제공하는 cvtColor함수의 내부적인 Grayscale 변환 방식은 다음과 같다.

Gray = Red \* 0.2126 + Green \* 07152 + Blue\*0.0722

따라서 이미지 stuff\_color.jpg 내에 존재하는 순수한 RGB로 이루어져 있는 각각의 원은 각자 다른 밝기로 변환된다. Green > Red > Blue 순으로 원의 밝기가 설정될 것이다. 변환된 Grayscale 이미지에서도 확인할 수 있다. 각자 변환된 Grayscale 이미지는 다음과 같다.

**Grayscale Image A GrayScale Image B**

1. **Threshold each image (Threshold type : CV\_THRESH\_BINARY)**

**(A\_binary, B\_binary)**

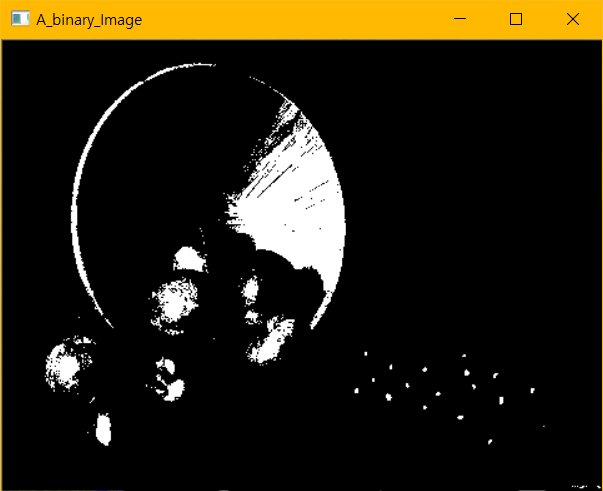
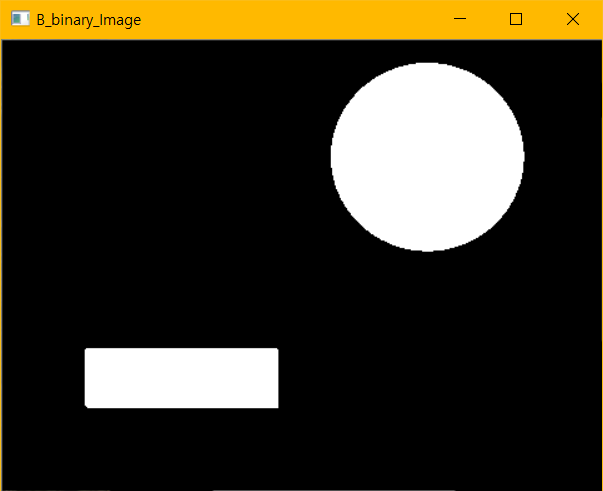
threshold 함수를 사용하여 이미지를 이진화한다. 이 때, 사용자가 적용하고 싶은 Thresholding Value를 직접 설정해줄 수 있다. 3번째 매개변수로서 적용하고자 하는 Thresholding Value를 넣을 수 있고, 4번째 매개변수로서 Thresh holding Value를 넘어서는 값들의 밝기 값을 설정해줄 수 있다. 여기서는 0과 255로만 이루어진 이진 영상으로 설정할 것이므로 매개변수로 255를 주었다. 마지막 매개변수 옵션은 이진 변환을 뜻하는 것이다. 따라서 그림의 127 밝기 미만의 픽셀은 모두 0의 밝기를 가지고 127밝기 이상의 픽셀은 255의 밝기를 가지게 된다. 이미지를 변환하는 코드는 다음과 같다.

**//Convert Images with threshold value 127 into Binary Image**

**double binary\_value\_A = threshold(img1\_gray, img1\_binary, 127, 255, CV\_THRESH\_BINARY);**

**double binary\_value\_B = threshold(img2\_gray, img2\_binary, 127, 255, CV\_THRESH\_BINARY);**

위의 두 이미지를 Thresholding Value 127 로 이진 영상으로 변환하면 다음과 같다.

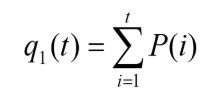
**Binary Image A Binary Image B**

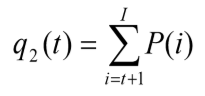
**Image B의 경우 Green 밝기 값을 제외한 Green Red 밝기 값들이 모두 127 아래의 값이였기 때문에 0으로 변환된 것을 확인할 수 있다.**

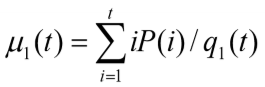
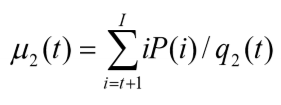
1. **Threshold each image (Threshold type : CV\_THRESH\_BINARY | CV\_THRESH\_OTSU)**

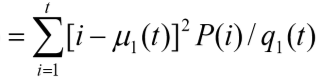
**(A\_otsu, B\_otsu)**

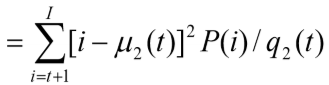
Otsu Method를 사용하면 이진화 영상에 적정한 thresholding value를 알아서 정할 수 있다. 따라서 사용자는 임의의 thresholding값을 따로 정해줄 필요가 없으며 적정한 thresholding value를 정할 수 있는 공식은 다음과 같다.

q1(t)는 Thresholding Value t 미만의 픽셀들의 비율을 모두 더한 값이다.

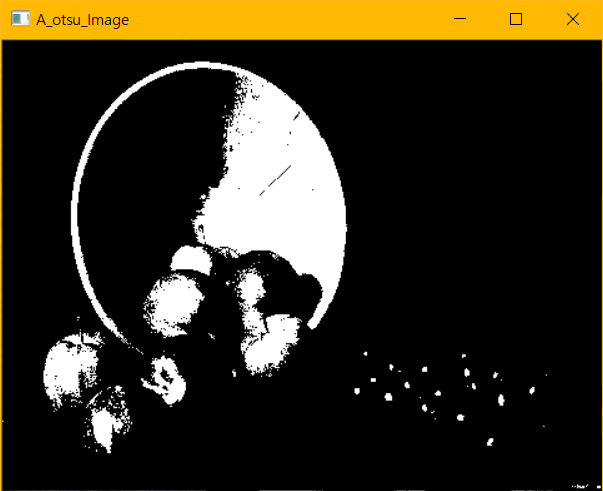
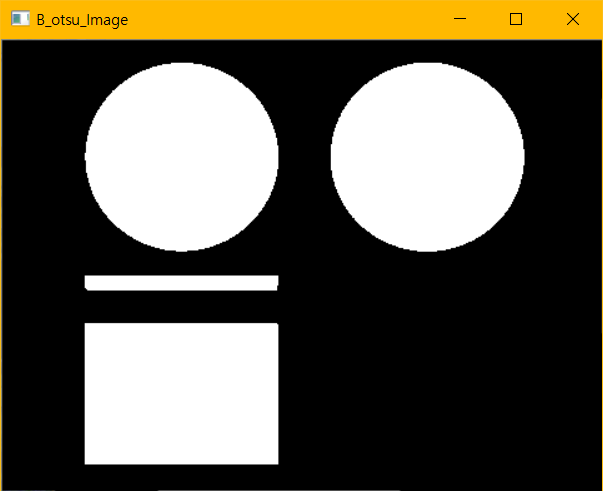
 q2(t)는 Thresholding Value t 이상의 픽셀들의 비율을 모두 더한 값이다.

  이렇게 u 두 값을 각각 구할 수 있다.

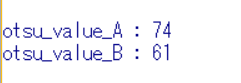
  Thresholding Value t 미만의 값들의 분산값

  Thresholding Value t 이상의 값들의 분산값

 다음 값은 해당 이미지의 모든 값들의 총 분산이다. 나올 수 있는 분산 값들 중 가장 작은 값을 선택한다. 이 값이 Otsu Method에 의해 정해진 Thresholding Value이다. 코드는 다음과 같다.

**Otsu\_Image A Otsu\_Image B**



**Otsu Method에 의해 결정된 Thresholding Value**

**Image B의 Red 값은 74보다 밝은 값이기 때문에 위의 Binary Image와 달리 255의 밝기를 가지는 것을 확인할 수 있다. 네모 영역도 대부분 Binary Image와 다르게255의 밝기를 가지고 있는 것을 확인할 수 있다. 코드는 다음과 같다.**

**//Convert Images with Otsu Method Binarization**

**double otsu\_value\_A = threshold(img1\_gray, img1\_otsu, 0, 255, CV\_THRESH\_OTSU);**

**double otsu\_value\_B = threshold(img2\_gray, img2\_otsu, 0, 255, CV\_THRESH\_OTSU);**

1. **Adaptive threshold each image (Adaptive method: ADAPTIVE\_THRESH\_MEAN\_C, Threshold type : CV\_THRESH\_BINARY)**

**(A\_adaptive, B\_adaptive)**

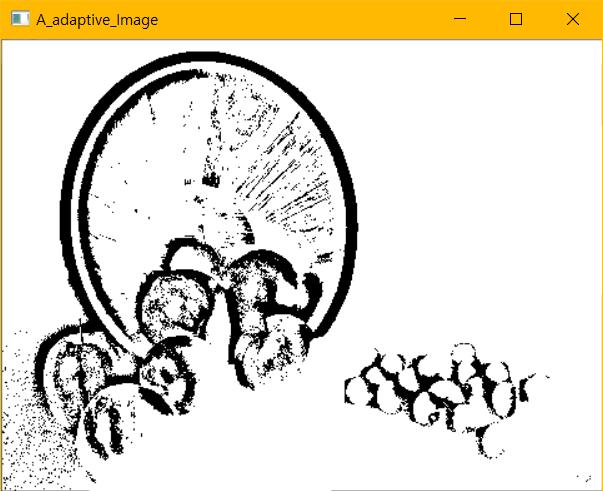
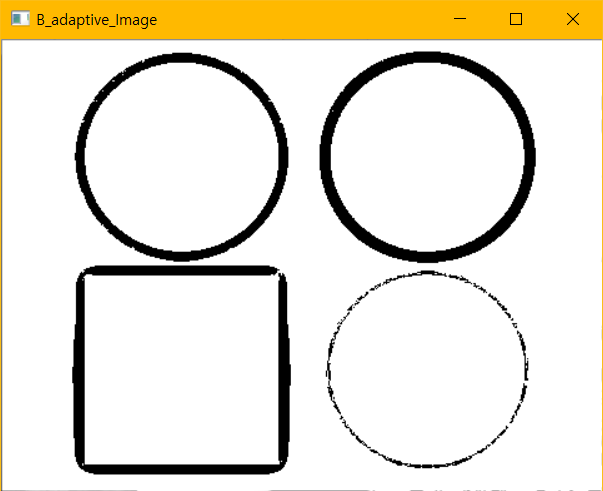
**Adapative Threshold는 하나의 thresholding value만 사용하는 다른 이진화 기법과 다르게 주변의 픽셀 밝기에 따라서 각자 픽셀에 다른 thresholding value를 적용하는 기법이다. Adaptive Threshold는 다음과 같은 adaptiveThreshold 함수로 다음 코드와 같이 구현한다. 이 때 3번째 매개변수는 Adaptive 함수를 적용하는 옵션을 뜻한다. 5번째와 6번째 매개변수는 각각 픽셀에 인접하여 thresholding value에 영향을 주는 블록의 범위와 인접 픽셀 비교 차이값을 뜻한다.**

**//Convert Images with Adaptive Binarization**

**adaptiveThreshold(img1\_gray, img1\_adaptive, 255, CV\_ADAPTIVE\_THRESH\_MEAN\_C, CV\_THRESH\_BINARY, 21, 10);**

**adaptiveThreshold(img2\_gray, img2\_adaptive, 255, CV\_ADAPTIVE\_THRESH\_MEAN\_C, CV\_THRESH\_BINARY, 21, 10);**

**위와 같이 적용한 이미지는 다음과 같다.**

**A\_adaptive\_Image B\_adaptive\_Image**

**픽셀값의 변화에 따른 경계점을 기준으로 0의 밝기가 형성 되어있는 것을 확인할 수 있다.**

1. **Open the project folder to check output images**

**-Compare the results of above steps**

**다음 코드로 이미지들을 모두 저장하였다.**

**//Store Binary Images**

**imwrite("A\_binary\_Image.jpg", img1\_binary);**

**imwrite("B\_binary\_Image.jpg", img2\_binary);**

**//Store Otsu Binary Images**

**imwrite("A\_otsu\_Image.jpg", img1\_otsu);**

**imwrite("B\_otsu\_Image.jpg", img2\_otsu);**

**//Store Adaptive Binary Images**

**imwrite("A\_adaptive\_Image.jpg", img1\_adaptive);**

**imwrite("B\_adaptive\_Image.jpg", img2\_adaptive);**