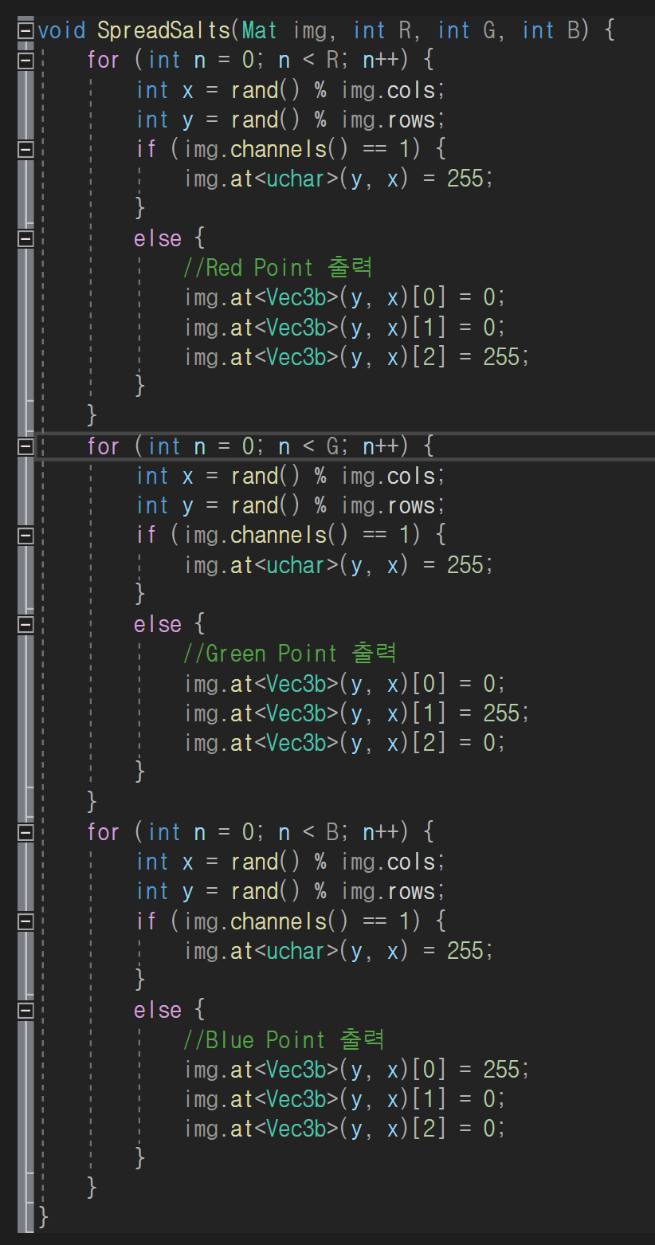
3nd Week Lab Assignment

12201928 이상혁

**# Task 1**

▪ 주어진 영상(img1.jpg)에 빨강, 파랑, 초록 색의 점을 각각 설정한 개수만큼 무작위로 생성하는 프로그램을 작성할 것

**-void SpeardSalts**

4개의 매개변수를 받는 SpreadSalt 함수를 만들었다. R,G,B 각각의 점 개수를 매개변수로 받았으며 rand에 픽셀 값을 % 계산하여 사진 내에 랜덤으로 찍힐 수 있게 구현하였다.  
[0] 이 BLUE, [1] 이 Green, [2] 가 Red를 담당하므로 각각의 배열의 값에 255을 대입하고 다른 값들에는 0을 대입하여 온전한 색이 나올 수 있도록 하였다.

**-main**

코드를 실행할때마다 랜덤으로 출력될 수 있도록 srand를 사용하였다.

Imread를 사용해 img1을 불러왔으며 RGB 각각 30,50,70개 점이 랜덤으로 찍히도록 함수에 매개변수로 전달해주었다.

점이 찍힌 다음 imshow로 이미지 출력을 진행하였다. 키입력대기, 이미지 출력종료까지 실행한다.

**Figure 1 void SpeardSalts 함수Code 사진**

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명

**Figure 2 main함수Code 사진**

하늘, 구름, 야외, 로켓이(가) 표시된 사진

자동 생성된 설명구름, 하늘, 미사일, 야외이(가) 표시된 사진

자동 생성된 설명

**Figure 3 좌 – 원본사진, 우- 점을 찍은 후 사진**

**# Task 2**

▪ 앞서 생성한 영상에서 빨강, 파랑, 초록 색의 점을 각각 카운트하는 프로그램을 작성하고 카운트 결과가 실제와 일치하는지 검증할 것

스크린샷, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**Figure 4 점의 개수를 새는 Count 함수 CODE**

1. for 루프를 이용하여 이미지의 모든 열과 행을 반복한다.
2. 각 픽셀에 대해 해당하는 RGB 색상 값이 특정 값인지 확인한다.

현재 픽셀의 색상이 (0, 0, 255)인 경우, R\_P 변수를 증가,

현재 픽셀의 색상이 (0, 255, 0)인 경우, G\_P 변수를 증가

현재 픽셀의 색상이 (255, 0, 0)인 경우, B\_P 변수를 증가

그 후 저장된 값들을 출력하여 확인할 수 있도록 하였다. main에서는 이 함수를 사용하는 코드만 작성하였다.

다음 페이지의 사진에서 확인할 수 있듯이 위 Task1에서 보낸 값인 30,50,70이 동일하게 찍힘을 확인할 수 있었다.

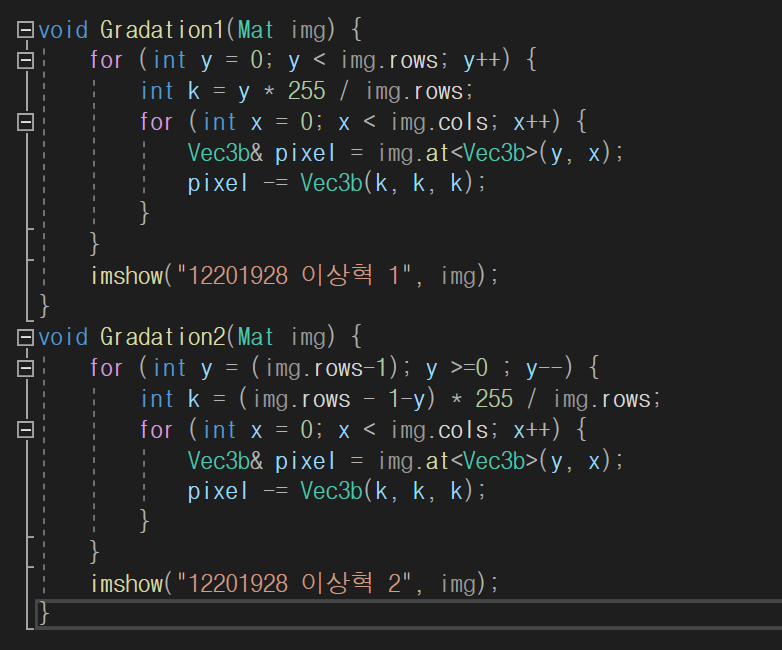
텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명텍스트, 구름, 하늘, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**Figure 5 좌 – Count함수를 실행한 main CODE,   
우 – Figure 3의 사진에서 점을 COUNT 한 후 출력된 R,G,B의 개수**

**# Task 3**

▪ 주어진 영상을 이용해(img2.jpg) 다음과 같은 두 영상을 생성하는 프로그램을 작성하고(픽셀 값 접근을 이용) 히스토그램 일치 여부를 확인 및 그러 한 결과가 나온 이유를 분석할 것

**Figure 6  Task3에서 두 영상을 생성하는 함수 CODE**

Gradation 1

이미지의 각 행에 대해 반복하면서, 그 행의 위치에 그라데이션을 적용하게 된다.   
y값을 기준으로 0에서 img.rows까지의 범위에서 0부터 255까지의 값을 생성하고, 이를 k에 저장한다. 그리고 이미지의 각 픽셀에 대해 현재 픽셀의 각 채널 값에서 k만큼 빼준다.

이렇게 진행을 하게 되면 이미지의 아래쪽으로 갈수록 어두워지는 그라데이션 효과가 적용된다.

Gradation 2

이미지의 각 행에 대해 반대 즉 역순으로 반복하면서, 그 행의 위치에 그라데이션을 적용하게 된다. y값을 기준으로 img.rows-1에서 0까지의 범위에서까지의 범위에서 0부터 255까지의 값을 생성하고, 이를 k에 저장한다. 그리고 이미지의 각 픽셀에 대해 현재 픽셀의 각 채널 값에서 k만큼 빼준다.

이렇게 진행을 하게 되면 이미지의 위으로 갈수록 어두워지는 그라데이션 효과가 적용된다.

이후 동일하게 imshow을 사용하여 이미지를 표시한다.

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**Figure 7  Task3의 Main CODE**

텍스트, 스크린샷, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**Figure 8  코드를 실행시킨 결과 왼쪽이 Gradation1 오른쪽이 Gradation2 이다.**

**텍스트, 스크린샷, 디스플레이이(가) 표시된 사진

자동 생성된 설명**

**Figure 9  히스토그램을 진행할 코드**

텍스트, 스크린샷, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**Figure 10  히스토그램 사진**

거의 유사한 모습의 histogram을 확인할 수 있다.  화소의 강도(Gray 값 또는 하나의 채널에 대한 값)를 갖는 화소의 개수를 각각 X축과 Y축으로 표시한다. 오른쪽의 히스토그램이 조금 더 화소의 강도가 높음을 확인할 수 있다. 이미지에도 보면 알 수 있듯이 오른쪽 이미지가 조금더 밝기가 높기 때문으로 분석할 수 있다. 그 이외의 값들은 비슷한 채널의 color로 이루어져 있으며 비슷한 화소의 개수 및 밝기로 인해 유사하다고 판단할 수 있었다.

**# Task 4**

▪ 주어진 영상(img3.jpg, img4.jpg, img5.jpg)을 이용해 다음의 영상을 완성할 것

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

**Figure 11  주어진 영상을 만들기 위한 함수**

MakeVideo 함수는 세 개의 이미지(img3, img4, img5)를 사용한다.

1. Resize를 사용해서 img4 이미지를 img3 이미지와 동일한 크기로 조정하게 된다, 이때 size로 im3과 동일한 크기를 가지도록 조정한다.
2. Subtract를 사용하여 두 이미지의 histogram을 빼주게 된다. 이렇게 한다면 아래의 사진과 같이 img4의 밝은 부분이 img3에서 어둡게 나타날 수 있게된다!
3. 이후 spaceX 로고가 담긴 img5의 size를 resize 해준다.
4. Cvtcolor을 사용하여 grayscale 영사으로 변환해준다.
5. 이후 threshold 함수를 사용하여 img5\_gray를 임계값 180을 기준으로 이진화한 이후 mask에 저장하게 됩니다. 이때 임계값 조절에 따라서 mask의 생성 모습 차이를 확인할 수 있었다.
6. Rect 클래스를 사용하여 img5를 적용할 Roi를 만들어주었다  
   여기서는 (320, 340) 위치에서 img5의 크기로 해주었다.
7. img5를 result1의 ROI에 적용하여 결과를 result1\_roi에 저장한다. 이 때, img5\_mask를 마스킹하여 적용한다.
8. 이후 copyto를 사용하여 로고 마스크와 만들어준 roi를 합쳐준다.

스크린샷, 지상, 야외이(가) 표시된 사진

자동 생성된 설명

**Figure 12  MakeVideo를 실행한 결과**

**고찰 : openCV에 대한 여러가지 기능을 다루어보면서 배우고 이에 주어진 과제를 실행해보았다. 픽셀 값 접근에 대해서 이해할 수 있었으며 matrix operation 인 subtract 연산을 해볼 수 있을 수 있었다. 마지막 과제에서 어려움이 있었다. 처음에는 그냥 합쳤는데 배경이 제거 되지 않은 상태였기에 덮어버리는 모습이였다. 이를 해결하기위해 배경제거 방법을 찾게 되었고, 이진화와 마스크를 이용한 배경제거로 글자만 뽑을 수 있었다. 그 이후 roi에 대해서 알 수 있었으며 적절한 값의 할당이 중요함을 확인하게 되었다.**