REPORT

과목명	영상처리
담당교수	김동근 교수님
학과	컴퓨터공학부 컴퓨터공학전공
학번	201401932
이름	노 명 환

목 차

1. 서론

최근 온라인 동영상 서비스인 Youtube가 크게 성장하면서 동영상 시장이 크게 성장하고 있다. Youtube는 전문적인 동영상이 아니더라도 누구나 자신이 원하는 동영상을 업로드 할 수 있기 때문에 이에 관심 있는 많은 사람들이 스스로 동영상을 편집하여 온라인에 게시하고는 한다.

이러한 현상에 발맞춰 자연스럽게 규모가 커지는 채널은 스스로 컨텐츠를 제작하며 동영상 편집을 전문 인력에게 맡기는 외주 형식으로 변화하여가고 있으며 크리에이터도 점차 소규모 회사의 형태를 갖춰나가고 있다.

이처럼 누구나 동영상을 편집하고 온라인으로 게시할 수 있는 상태에서 자연스럽게 사람들은 동영상을 보다 쉽게 편집하기 위하여 다양한 툴을 찾고 사용하고자 한다. 이러한 수요를 해결하고 충족하기 위해서는 보다 쉽고 강력한 소프트웨어가 필요하다. 추후에 이러한 소프트웨어를 개발하기 위하여 이번 영상처리 수업은 영상을 편집하거나 개선하는 내부의 동작과정을 직접 구현하고 동작시키는 것은 큰 도움이 되리라 생각한다.

2. 본론

2.1 Mean Filter

중간값 필터는 비선형 디지털 필터 기술로 이미지나 기타 신호로부터 신호 잡음을 제거하는 데 자우 이용되며, 이미지 프로세싱에서 윤곽선 감지 같은 처리를 수행하기 전 단계인 이미지에 잡음 제거 등을 수행하는데 이용된다.

주변에 위치한 픽셀들을 픽셀 크기만큼 가져와서 해당 픽셀 값을 주변 픽셀 값들의 평균으로 변경시킨다. 이러한 작업은 전체적인 픽셀의 값의 변화가 작아지는 효과를 주므로 이미지의 잡음을 줄이는데 도움을 주게 된다.

2.2 Weighted Mean Filter

중간값 필터가 주변에 위치한 픽셀에서 모든 값을 평등하게 반영하였다면 가중 평균 필터에서는 주변 픽셀 값들에 대하여 가중치를 부여하였다. 가장 중앙에 위치한 자신에게 가장 높은 가중치를 부여하며 주변에 위치한 필터들에 대해서는 거리에 따라 차등하게 가중치를 부여함으로써 보다 중간값 필터보다는 보다 원본에 가까운 이미지를 얻을 수 있다.

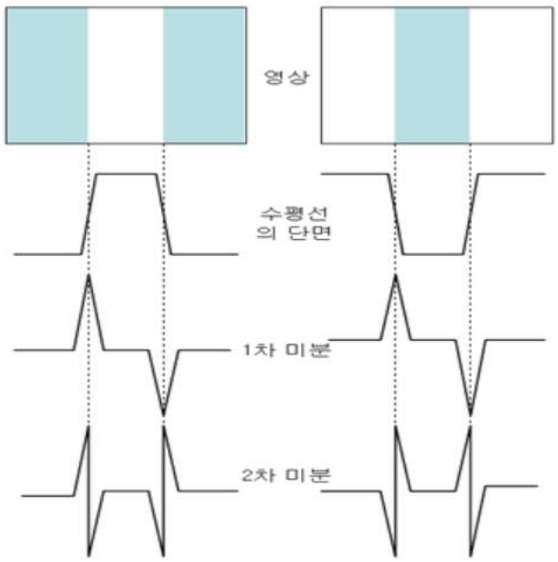
2.3 Sobel Filter

Sobel Operator는 주로 edge detection을 하는 데 사용하며 미분을 통하여 변화율을 측정한다. x축과 y축에 대하여 편미분을 함으로써 각각에 대한 변화율을 얻을 수 있다. 필터의 사이즈만큼 인접한 픽셀들에 대하여 변화율을 얻을 수 있으며 이를 종

합하면 edge를 검출할 수 있게 된다. 또한 단순히 x와 y뿐만이 아니라 대각 edge에 대한 검출도 가능하다.

2.4 Laplacian Filter

등방성 미분 연산자로써 이차미분을 수행하며 이미지가 변화하는 부분을 강조시켜주거나 edge detection을 하는 데 사용된다. edge detection을 수행할 때는 zero crossing이라는 과정을 수행하게 되는데 이차 미분 값의 부호는 edge 화소가 edge의 밝은 부분에 있는지 어두운 부분에 있는지를 결정하는데 사용된다. 이러한 특성을 이용하여 zero crossing에서는 부호의 변화를 판별하여 해당 부분이 edge를 판별하게 된다.



3. 실험결과

모든 실험은 학교에서 촬영한 8공학관 및 9공학관이 촬영되어 있는 사진을 편집하여 사용하였다.

[Original Image]

[Original Image with grayscale]

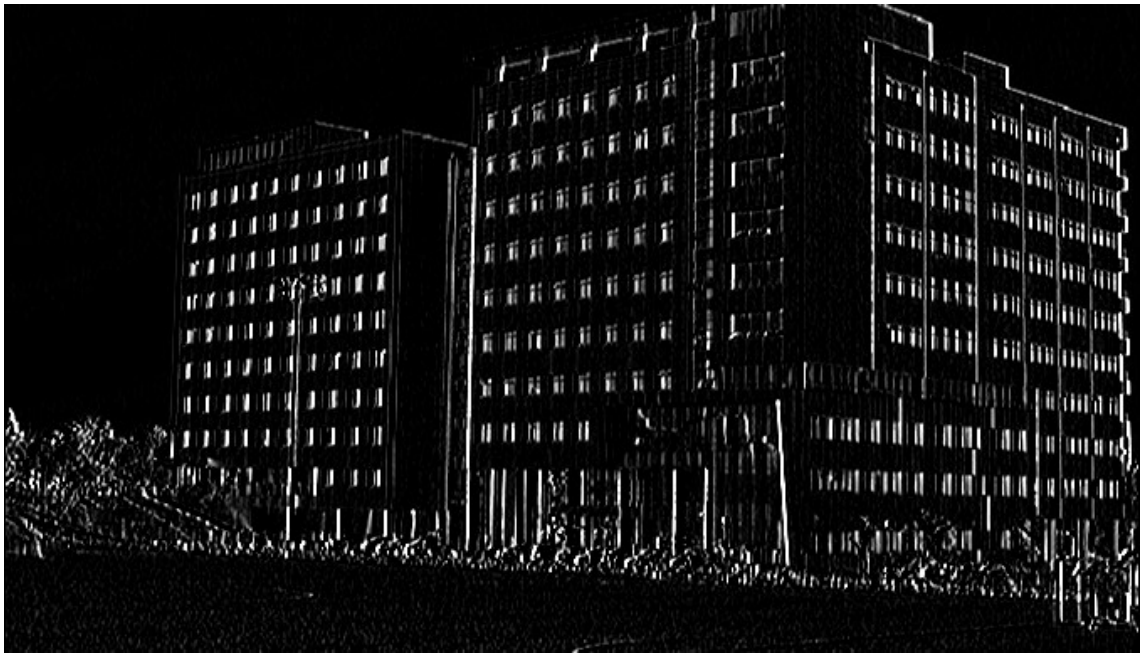
3.1 Mean Filter

원본과 비교하였을 때 사진이 전체적으로 부드러워 보이는 느낌을 준다. 이러한 이유는 평균 필터를 적용하였을 시 인접한 픽셀들에 대하여 평균값을 취하기 때문에 각 인접한 픽셀 값들의 변화가 크지 않기 때문이다.

3.2 Weighted Mean Filter

부드러운 느낌이지만 Mean Filter와 비교하였을 때는 중앙에 가까운 픽셀에 대하여 보다 큰 가중치를 주었기 때문에 조금의 변화가 나타난다.

3.3 Sobel Filter

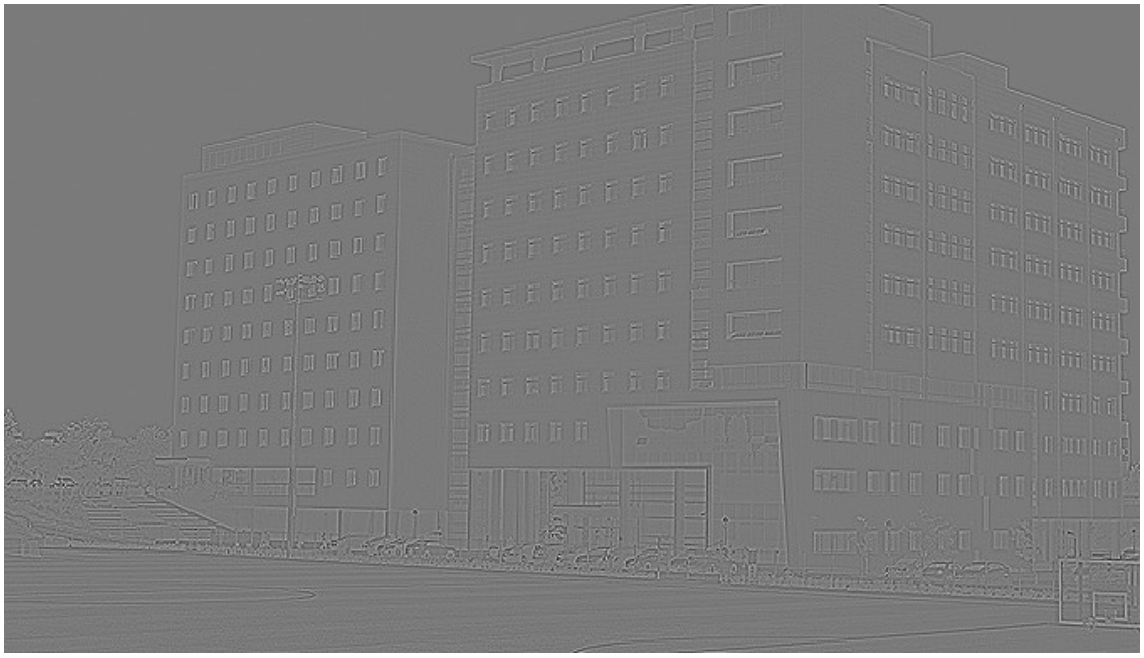


Sobel Filter의 X에 대한 결과로 x축에 대하여 미분을 진행하였기 때문에 y축 즉 수직에 대한 성분이 남아있는 것을 확인할 수 있다. 필터를 통하여 인접 픽셀에 미분을 하게 되었을 때 밝기 변화율이 기울기의 형태로 나타나게 되는데 기울기가 크다는 것은 밝기 값의 변화가 크다는 의미이므로 영역의 경계면으로 edge가 된다.

Sobel Filter의 Y에 대한 결과로 마찬가지로 y축에 대하여 미분을 진행하였기 때문에 x축 성분이 남아있다.

x, y에 대한 연산 결과를 magnitude 함수의 결과로 edge가 훌륭하게 검출 된 것을 확인 할 수 있다.

3.4 Laplacian Filter

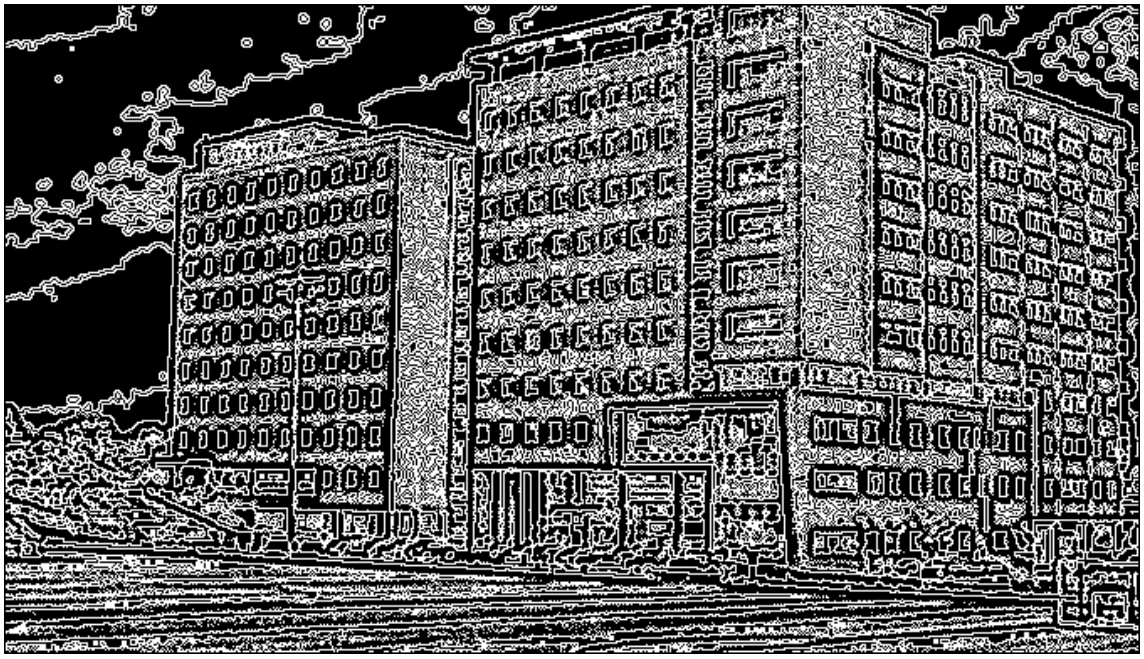


Laplacian Filter를 수행한 결과로써 이차 미분을 수행하였기 때문에 얇은 선이나 점 같은 고립된 부분에 대하여 강한 응답을 가진다. 또한 일차 미분을 수행한 Sobel Filter와 비교하였을 때 선이 보다 얇으며 적은 픽셀에 표현이 된 것을 확인할 수 있다.

Laplacian Filter가 적용 된 그림에 대하여 zeroCrossing을 적용한 결과이다. 그림을 보았을 때 상당히 복잡하고 난해한 결과를 보이는 것을 확인할 수 있는데 이에 대하여 2가지의 가설을 세우게 되었다.

첫째, 해당 실험에서는 3x3이라는 작은 필터 크기를 적용하였는데 이 때문에 각 픽셀에 대하여 민감하게 반응을 하였다.

둘째, 별도의 blur를 적응하지 않아 각 픽셀 사이 값들의 변화 값이 크기 때문에 거의 모든 픽셀에 대하여 zero crossing이 검출되었다.



Gaussian Blur를 적용한 이후 zero crossing을 수행한 결과이다. 확실히 이전과

비교하였을 때 전체적으로 이미지가 훨씬 깔끔하고 직관적으로 다가오는 것을 확인할 수 있었다.

원본 이미지에서 Laplacian filter를 적용한 이미지를 subtract한 이미지이다. Laplacian filter는 단순히 edge를 판별하는 것만이 아니라 원본 이미지와 연산을 통하여 보다 선명도가 높은 이미지를 얻을 수 있다는 장점을 가지고 있다. 해당 이미지와 원본 이미지를 비교해보면 이미지가 보다 선명해 보인다는 것을 알 수 있다.

4. 결론

다양한 filter를 직접 정의한 이후 filter2D 함수를 이용하여 적용시켜본 결과, 이미지가 어떠한 과정을 거쳐 처리가 되는지에 대한 흐름과 이미지를 편집할 때 스케치 효과 등 다양한 효과 등이 어떤 방식으로 이루어지는지 알 수 있었다. 이러한 과정을 직접 구현해봄으로써 이에 관련하여 더욱 이해가 깊어진 것 같다.

핸드폰의 앱스토어를 보면 다양한 카메라 어플리케이션과 카메라에 사용할 수 있는 필터가 존재한다. 이러한 카메라 어플리케이션은 대중들의 선호를 받을 때 편의성도 중요시 여기지만 보다 크게 각광받는 것은 다름 아닌 필터이다. 실제로 애플스토어에서 유료 순위를 확인하면 다양한 카메라 필터들이 상위권에 위치하고 있는 것을 확인할 수 있으며 다양한 필터를 제공하는 어플리케이션일수록 사람들의 선호를 받는 것을 확인할 수 있다. 이러한 다양한 filter와 blur를 이용하여 카메라 어플리케이션을 개발하면 추후에 큰 사업아이템이 될 수 있을 것 같다는 생각이 들었다.

추가로 이번 과제를 진행하면서 교재에서 오류를 발견하였다. 예제 6.6과 6.8에 구현되어 있는 zeroCrossing 함수에서 width, height = lap.shape 라는 코드가 있는데 이 때, shape 함수에서 return해주는 parameter는 y축, x축의 순서로 이루어지기 때문에 height, width = lap.shape가 옳은 표현이다. 해당 예제에서는 lena.jpg라는 이미지를 통하여 해당 함수가 실행되는데 이 이미지는 512x512의 height와 width의 크기가 같기 때문에 정상적으로 동작하지만 실제로 다른 이미지를 input으로 넣게 될 경우 오류가 발생한다.

5. 참고문헌

[Mean Filter]
https://ko.wikipedia.org/wiki/%EC%A4%91%EA%B0%84%EA%B0%92_%ED%95%84%ED%84%B0
[Sobel Operator]
https://en.wikipedia.org/wiki/Sobel_operator
https://bskyvision.com/43
[Laplacian Filter]
https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm
http://blog.daum.net/trts1004/12109067
[Zero Crossing]
https://en.wikipedia.org/wiki/Zero_crossing

6. 부록

[Source Code]

```
import cv2
import numpy as np

#Read image with bgr and grayscale
src  = cv2.imread('./school.png')
src_gray  = cv2.imread('./school.png', 0)

#1 Mean Filter
mean = [[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]]
mean = np.array(mean) / 9
mean_result = cv2.filter2D(src, cv2.CV_32F, mean)
mean_result = cv2.normalize(mean_result, None, 0, 255, cv2.NORM_MINMAX, dtype
```

```
= cv2.CV_8U)

#2 Weighted Mean Filter
wMean = [[1, 2, 1],
         [2, 4, 2],
         [1, 2, 1]]
wMean = np.array(wMean) / 16
wMean_result = cv2.filter2D(src, cv2.CV_32F, wMean)
wMean_result = cv2.normalize(wMean_result, None, 0, 255, cv2.NORM_MINMAX,
dtype = cv2.CV_8U)

#3 Sobel Filter
sobelX = [[-1, 0, 1],
          [-2, 0, 2],
          [-1, 0, 1]]
sobelX = np.array(sobelX)
sobelX_result = cv2.filter2D(src_gray, cv2.CV_32F, sobelX)

sobelY = [[-1, -2, -1],
          [0, 0, 0,],
          [1, 2, 1]]
sobelY = np.array(sobelY)
sobelY_result = cv2.filter2D(src_gray, cv2.CV_32F, sobelY)

mag = cv2.magnitude(sobelX_result, sobelY_result)
sobel_result = cv2.normalize(mag, None, 0, 255, cv2.NORM_MINMAX, dtype =
cv2.CV_8U)

#4 Laplacian Filter
def getSign(number):
    if number >= 0:
        return 1
    else:
        return -1

def zeroCrossing(mat):
    height, width = mat.shape
    result = np.zeros(mat.shape, dtype = np.uint8)
    for y in range(1, height - 1):
        for x in range(1, width - 1):
            neighbors = [mat[y-1,x], mat[y+1,x], mat[y,x-1], mat[y,x+1],
```

```
                        mat[y-1,x-1],    mat[y-1,x+1],    mat[y+1,x-1],    mat[y+1,x+1]]

            minimum = min(neighbors)
            if getSign(mat[y,x]) != getSign(minimum):
                result[y, x] = 255
    return result

laplacian = [[0, 1, 0],
             [1, -4, 1],
             [0, 1, 0]]
laplacian = np.array(laplacian)

laplacian2 = [[0, -1, 0],
             [-1, 5, -1],
             [0, -1, 0]]
laplacian2 = np.array(laplacian2)

blur = cv2.GaussianBlur(src_gray, ksize = (7, 7), sigmaX = 0.0)
laplacian_apply = cv2.filter2D(src_gray, cv2.CV_32F, laplacian)
laplacian_zeroCrossing = zeroCrossing(laplacian_apply)
laplacian_apply = cv2.normalize(laplacian_apply, None, 0, 255, cv2.NORM_MINMAX,
dtype = cv2.CV_8U)

laplacian_result = cv2.filter2D(src_gray, cv2.CV_32F, laplacian2)
laplacian_result = cv2.convertScaleAbs(laplacian_result)
laplacian_result = cv2.normalize(laplacian_result, None, 0, 255, cv2.NORM_MINMAX,
dtype = cv2.CV_8U)


#Show result
cv2.imshow('Original', src)
cv2.imshow('Mean Filter',  mean_result)
cv2.imshow('Weighted Mean Filter',  wMean_result)
cv2.imshow('Original with Grayscale', src_gray)
cv2.imshow('Sobel Filter X',  sobelX_result)
cv2.imshow('Sobel Filter Y',  sobelY_result)
cv2.imshow('Sobel Filter',  sobel_result)
cv2.imshow('Apply Laplacian Filter', laplacian_apply)
cv2.imshow('Laplacian Zero Crossing', laplacian_zeroCrossing)
cv2.imshow('Enhanced Image with Laplacian Filter', laplacian_result)
```

```
###Save the result
cv2.imwrite('./1_Original.png', src)
cv2.imwrite('./2_Mean Filter.png',  mean_result)
cv2.imwrite('./3_Weighted Mean Filter.png',  wMean_result)
cv2.imwrite('./4_Original with Grayscale.png', src_gray)
cv2.imwrite('./5_SobelX.png',  sobelX_result)
cv2.imwrite('./6_SobelY.png',  sobelY_result)
cv2.imwrite('./7_Sobel Filter.png',  sobel_result)
cv2.imwrite('./8_Apply Laplacian Filter.png', laplacian_apply)
cv2.imwrite('./9_Laplacian Zero Crossing.png', laplacian_zeroCrossing)
cv2.imwrite('./10_Enhanced Image with Laplacian Filter.png', laplacian_result)

cv2.waitKey()
cv2.destroyAllWindows()
```