

<영상처리#HW3 - 보고서>
학과 : 컴퓨터정보통신공학부
학번 : 2019xxxxxxx
이름 : 전xx

<1번 문제>

1번 문제의 요약

다양한 Edge-detection 방법 (로버츠, 피리윗, 소벨, 스토크스틱)을 이용하여 엣지 검출하고 노이즈가 추가된 이미지에서 각각의 성능을 비교 및 엣지의 수와 누락된 픽셀의 수 또는 새롭게 발견된 엣지 픽셀의 수를 비교하면서 감지 오류율을 계산하는 문제이며, 각각의 방법으로 노이즈가 추가된 이미지에서 얼마나 효과적으로 엣지를 감지하여 검출하는지에 대한 것을 평가하고 어느 방법이 특정 Signal-to-Noise Ratio에서 효과적인지를 확인하는 것이 목표인 문제입니다.

그림0-1. Lena_Original_Edge_Detection (150)



확장자가 RAW 파일 레나 사진을 비트맵 헤더를 이용하여 프린트한 레나 사진입니다.
이 사진을 통해 레나사진이 BMP형식으로 제대로 프린트 했다는 것을 알 수 있습니다.

그림 0-2. Lena_GaussianNoise_Edge_Detection (150)



bmp 사진으로 프린트된 레나 사진에 가우시안 노이즈를 추가한 사진입니다. 보시면 오리지널 사진과 달리 사진의 픽셀 값 곳곳에 가우시안 노이즈가 끼서 화질이 낮아 보이는 현상을 알 수 있습니다.

그림 1-1. Roberts_Original_Edge_Detection(150)



이 사진은 로버츠(Roberts) 방식으로 오리지날 사진을 엣지 검출한 것입니다. 사진을 보면, 윤곽선이 잘 안보이는 것을 알 수 있습니다.

그림1-1. Roberts_Edge_Detection(150)



이 사진은 가우시안 노이즈가 추가된 사진으로 엣지 검출한 사진입니다. 보시면 대략적인 윤곽선이 보이는 것을 알 수 있습니다.

로버츠 방식은 엣지 검출에 있어 사용되는 간단한 방법 중 하나이며, 두 개의 작은 마스크를 사용해, 이미지의 각 픽셀에 대한 수직 방향과 수평 방향의 강도 변화를 계산하는 방법입니다. 이러한 로버츠 방식의 장점은 간단하고 직관적인 방법이라 각 픽셀의 주변 강도 변화에 빠르게 계산하고, 아주 작은 디테일한 사항이나, 선명한 엣지를 감지하는데, 효과적인 방법입니다. 그러나 높은 주파수로 구성 요소를 감지하기 때문에, 잡음에 민감하고, 노이즈가 신호로 인식되어 잘못된 엣지를 생성할 수도 있습니다. 그리고 간단한 방법이라서 정확도가 낮습니다. 또한, 선명한 엣지는 잘 탐지하지만, 상대적으로 덜 선명한 엣지나 곡선 엣지에 대한 탐지에 대해 성능 검출이 떨어질 수도 있습니다.

그림 1-2. Prewitt_Original_Edge_Detection(150)



이 사진은 오리지날 사진에서 프리윗(prewitt)방식으로 뽑아낸 방식입니다, 오리지날 사진에서도 로버츠 방식보다 더 선명하게 엣지 검출 되었다는 것을 알 수 있습니다.

그림1-2. Prewitt_Edge_Detection(150)



가우시안 노이즈 사진에 프리윗(Prewitt)방식으로 검출한 사진입니다. 위에서 본 오리지날 프

리윗 방식과 비슷하게 엣지 검출이 되었으나 주변 잡음이 나타나 사진이 번지는 이미지로 보입니다 이는 가우시안 노이즈가 무작위한 값을 추가하여 이미지를 흐리게 만들었는데, 이를 엣지 검출 과정에서 불필요한 세부사항이나 잡음을 감지 할 수 있기 때문입니다

프리윗 마스크의 특징 - 소벨 마스크를 회전한 결과와 비슷하며, 응답시간이 빠릅니다, 소벨 마스크에 비해 밝기 변화에 대해 비중을 약간 적게 줘서 엣지 검출시 엣지가 덜 부각되는 현상이 있습니다, 대각선 방향 엣지보다 수직방향, 수평방향 엣지가 더 민감하게 반응합니다.

그림 1-3. Sobel_Original_Edge_Detection(150)



이 사진은 소벨 마스크 방식으로 검출한 사진이며, 오리지날 이미지의 프리윗 방식과 비슷하게 프린터 되었고 비교적 부드럽게 엣지가 검출 되었습니다.

그림 1-3. Sobel_Edge_Detection(150)



가우시안 노이즈를 추가한 이미지에서는 오리지널과 달리 각각의 불필요한 잡음들이 검출되었다는 것을 알 수 있습니다.

소벨 마스크 특징 - 수직 방향과 수평 방향에서의 밝기의 변화를 감지합니다. 이러한 방식은 각 픽셀의 주변의 밝기 변화를 파악하여, 해당 픽셀의 엣지를 검출 합니다. 소벨 엣지 검출은 각 픽셀의 주변 밝기 변화를 기반으로 엣지의 강도와 방향에 대한 정보를 제공하는 방식이며, 로버츠 필터에 비해 부드러운 엣지를 검출하기 때문에, 자연스러운 엣지 검출 결과를 얻을 수 있고, 필터의 구조가 간단하여 비교적 쉽고 효율적입니다. 하지만 엣지가 두꺼운 경향이 있기 때문에, 세부적인 엣지를 검출하지 못할 수도 있습니다. 엣지 주변에서의 노이즈나 불필요한 디테일을 검출할 수 있습니다.

그림 1-4. Stochastic_Original_Edge_Detection(150)



오리지날 이미지에서 5*5 스토캐스틱 방법으로 엣지를 검출한 사진입니다 비교적 선명하게 이미지가 검출되었고, 강한 엣지가 검출된 것을 알 수 있습니다.

그림 1-4. Stochastic_Edge_Detection(150)



가우시안 노이즈가 추가된 이미지를 스토캐스틱 마스크 엣지 검출 방법이며, 비교적 강한 엣

지검출을 하였지만, 특정 부분이나 픽셀의 값에 따라 잡음이 발생한걸 알 수 있습니다.

스토캐스틱 마스크 엡지 검출 방법은 그레디언트 기반의 엡지 검출 방법과 약간 다른 방법입니다. 하지만 스토캐스틱의 장점은 다른 엡지 검출 방법들보다 강한 엡지를 잘 감지하여 더 강한 엡지를 감지할 수 있습니다. 그리고 노이즈에 강해, 상대적으로 주변에 높은 노이즈가 있더라도, 비교적 정확한 엡지를 검출할 수 있습니다. 또한 피라매터를 조정하여 성능을 향상시킬 수 있기 때문에, 값을 조정하기에 쉬운편입니다. 하지만 노이즈나, 이미지의 특정 구조에 따라 일부 스토캐스틱들이 부정확한 엡지를 생성할 수 있습니다. 그리고 다른 엡지 검출에 비해 계산비용이 높아, 대규모 이미지나, 실시간 응용프로그램에서 사용이 어려울 수 있습니다. 장점으로 피라미터의 조정을하며 성능을 향상시킬수 있다고했지만, 역으로 피라미터의 값에 민감하여 알고리즘 설계시에 어려움이 있을 수도 있습니다.

<1번 문제 결론>

다양한 방식으로 엡지 검출을 해본 결과 각각의 장단점들이 있었으며, 각각의 에러비율을 계산 해본 결과, 로버츠는 4.0 내외, 프리윗은 1.0 내외, 소벨은 2.0 내외, 스토캐스틱은 0.5내외에서 에러 비율이 나오고 있습니다. 에러 비율이 낮을수록, 성능이 우수하고 노이즈에 덜 민감하고 정확한 이미지를 나타내는 것을 알 수 있기 때문에, 스토캐스틱, 프리윗, 소벨, 로버츠 순으로 성능이 좋다고 할 수 있습니다.

<2번 문제>

2번 문제의 요약

이 문제는 보트 이미지를 이용하여 노이즈가 추가된 이미지에 대해 3 * 3 저주파(Low-pass)와 중앙 값 (Mediran)의 필터를 비교하는 문제로 각 결과물의 MSE (평균 제곱 오차)를 얻어 비교하며, 노이즈 제거 및 이미지 복원에 더 효과적인 필터가 뭔지 확인하는 문제입니다.

그림 2-1. Lowpass_Original (전)



그림 2-1. Lowpass (후)



그림 2-2. Median_Original (전)



그림 2-2. Median (후)



<2번 문제>

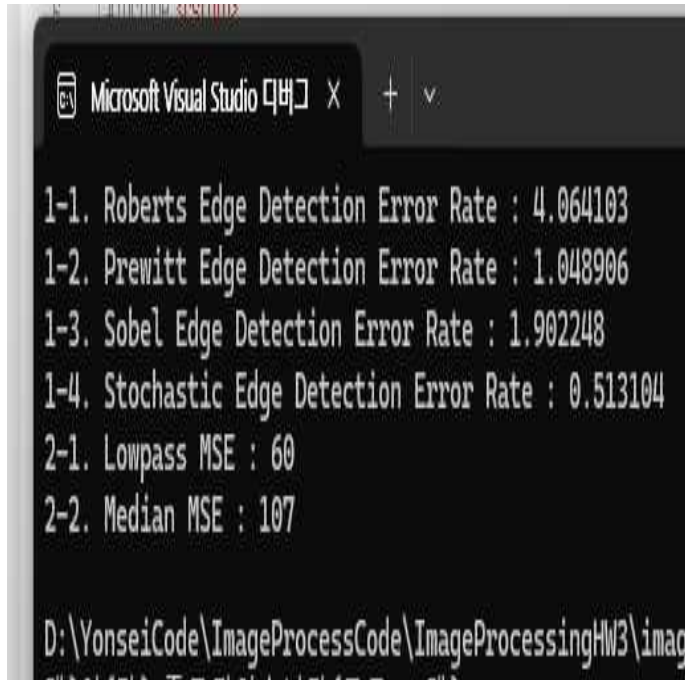
계산 결과 Low-pass는 60 값 내외(오차 있음)에서 MSE 값이 나오고 있고, Median의 값은 105 내외에서 결과가 나오기 때문에, 육안으로 비교 했을 때, 별차이 없어도, Low-pass 필터링이 더 좋은 성능이며 더 좋은 필터링 방법이라는 것을 알 수 있습니다. 그러한 이유는 MSE의 값이 작을수록 더 좋으며, MSE는 실제 이미지와 필터링된 이미지 간의 값의 차이를 나타내기 때문에, MSE의 값이 더 적을수록 필터링이 효과적이며, 원본 이미지에 가까운 값을 제공 한다는 것을 알 수 있습니다.

1,2번 문제 프로그램을 실행하고 나온 계산 결과 1

```
2019253084...처리#HW3.cpp X
imageProcessing_hw#3 (전역 범위)
1 // 영상처리 과제 3번
2 // 2019253084_전유성
3 #define _CRT_SECURE_NO_WARNINGS
4
5 #include <math>

Microsoft Visual Studio 디버그
1-1. Roberts Edge Detection Error Rate : 4.256410
1-2. Prewitt Edge Detection Error Rate : 1.067380
1-3. Sobel Edge Detection Error Rate : 1.928439
1-4. Stochastic Edge Detection Error Rate : 0.520007
2-1. Lowpass MSE : 59
2-2. Median MSE : 105
```

1,2번 문제 프로그램을 실행하고 나온 결과 2



```
Microsoft Visual Studio 디버그 X + v

1-1. Roberts Edge Detection Error Rate : 4.064103
1-2. Prewitt Edge Detection Error Rate : 1.048906
1-3. Sobel Edge Detection Error Rate : 1.902248
1-4. Stochastic Edge Detection Error Rate : 0.513104
2-1. Lowpass MSE : 60
2-2. Median MSE : 107

D:\YonseiCode\ImageProcessCode\ImageProcessingHW3\imag
```

<참조 문헌>

<https://blog.naver.com/bsw2428/221448393580> -네이버 블로그 영상처리 엣지 검출

<https://velog.io/@hihajeong/%EC%97%90%EC%A7%80%EA%B2%80%EC%B6%9C> - 엣지 검출

<https://junstar92.tistory.com/407> OpenCV를 이용한 엣지검출 (코드 Open Cv 사용 x, 이론만 참고)

강의자료 영상처리 3-2 ckarah