ReadMe

2376029 김문경

프로젝트 개요

이 프로젝트에서는 이미지 필터링을 C++와 OpenCV를 이용하여 구현했습니다. RGB 컬러 이미지와 grayscale 이미지에 대하여 Mean, gaussian, sobel, laplacian, 그리고 unsharp mask filtering 기법을 사용했습니다. 또한 tickmeter를 사용해 separable 방법으로 필터링을 적용했을 때와, 일반적인 방법으로 필터링을 적용했을 때의 소요 시간을 비교했습니다.

제출 파일 구성

파일 이름	설명
MeanFilterGray.cpp	Grayscale 이미지에 Mean filtering 적용
MeanFilterRGB.cpp	RGB 이미지에 Mean filtering 적용
GaussianGraySkeleton.cpp	Grayscale 이미지에 Gaussian filtering 적용
GaussianRGB.cpp	RGB 이미지에 Gaussian filtering 적용
SobelGraySkeleton.cpp	Grayscale 이미지에 Sobel filter 적용
SobelRGB.cpp	RGB 이미지에 Sobel filter 적용
laplacianGray.cpp	Grayscale 이미지에 laplacian filter 적용
laplacianRGB.cpp	RGB 이미지에 laplacian filter 적용
GaussianGraySep.cpp	Separable Gaussian 필터 구현 in GrayScale
GaussianRGBsep.cpp	Separable Gaussian 필터 구현 in RGB
Unsharp.cpp	Grayscale 이미지용 Unsharp Mask 구현
UnsharpRGB.cpp	RGB 이미지용 Unsharp Mask 구현

How to Run

1. OpenCV 4.11.0 version을 다운로드 받기

2. C++ 컴파일러를 이용해 파일 컴파일
3. Run 시키기

   (소요 시간의 경우, 디버그 창 확인)

Algorithm Summary

Mean filter

- 구현 방식
  - Mat meanfilter(const Mat input, int n, const char* opt);
  - kernel = Mat::ones(kernel_size, kernel_size, CV_32F) / (float) (kernel_size * kernel_size);

Gaussian filter

- 구현 방식
  - Mat gaussianfilter(const Mat input, int n, float sigmaT, float sigmaS, const char* opt);
  - kernel → gaussian 공식에 맞춰 계산

Sobel filter

- 구현 방식
  - Mat sobelfilter(const Mat input);
  - kernel

```
int Sx[3][3] = { {-1, 0, 1},
                 {-2, 0, 2},
                 {-1, 0, 1} };

   int Sy[3][3] = { {-1, -2, -1},
```

```
                { 0,  0,  0},
                { 1,  2,  1} };
```

Laplacian filter

- 구현 방식
  - Mat laplacianfilter(const Mat input);
  - kernel

```
int L[3][3] = { {0, 1, 0},
                {1, -4, 1},
                {0, 1, 0} };
```

Gaussian filter (Seperable ver.)

- 2D Gaussian 필터를 X, Y 방향으로 나누어 연산 효율 향상 ($O(n^2)$ → $O(2n)$)
- 구현 방식
  - Mat gaussianfilter(const Mat input, int n, float sigmaT, float sigmaS, const char* opt);
  - kernel → gaussian 공식에 맞춰 계산

Unsharp Masking

- 원본 이미지에서 Gaussian Blur 이미지를 빼고, 강조 비율 `k`에 따라 선명도를 조절
- 구현 방식
  - Mat UnsharpMask(const Mat input, int n, float sigmaT, float sigmaS, const char* opt, float k);
  - kernel → gaussian 공식에 맞춰 계산

경계 처리 방법

- zero-paddle
  - 이미지 밖은 0으로 간주
- mirroring
  - boundary를 이미지 내부로 반사
- adjustkernel
  - 유효 커널 영역만 정규화