미니 프로젝트 (Version 1.0)

# AI 비전 활용 공정제어 실무

프로젝트 제목 : OPENCV 없이 구현한 C언어 DIGITAL IMAGE PROCESSING

발표자 : 김대환

https://github.com/000526KDH/Ai_Gray_Scale

# 목차

# 프로젝트의 개요

- 이 프로젝트는 C 언어를 사용하여 회색조 이미지를 처리하는 다양한 영상 처리 기능을 구현하는 콘솔 애플리케이션입니다. Windows API를 활용하여 이미지를 불러오고, 처리된 결과를 화면에 표시하는 기능을 포함합니다.

# 개발환경

- **프로그래밍 언어**:
  - C 언어
- **운영 체제**:
  - Windows (특히, Windows 10 및 Windows 11)
- **필요한 라이브러리 및 헤더 파일**:
  - `stdio.h`: 표준 입출력 함수 사용
  - `Windows.h`: Windows API 사용을 위한 헤더
  - `math.h`: 수학 함수 사용

# 프로그램의 특이점

- **다양한 영상 처리 기능**
- **밝기 조절 및 반전**: 이미지의 밝기를 조절하고 색상을 반전시키는 기능을 제공합니다.
- **확대 및 축소**: 단순한 픽셀 복제 확대와 선형 보간법을 사용한 고품질 확대를 모두 지원합니다.
- **회전**: 사용자 입력 각도로 이미지를 회전시키는 기능을 두 가지 방식으로 제공합니다.
- **히스토그램 처리**: 히스토그램 스트레칭, 엔드-인 탐색, 히스토그램 평활화와 같은 고급 명암비 조정 기능을 포함합니다.
- **엠보싱 및 블러링**: 이미지에 엠보싱 효과와 블러 효과를 적용할 수 있습니다.
- **엣지 검출**: 여러 엣지 검출 알고리즘(Sobel, Laplacian 등)을 통해 이미지의 경계선을 강조할 수 있습니다.

## 프로그램의 주요 흐름

- 프로그램 실행 후 콘솔에 메뉴가 표시됩니다.
- 사용자는 메뉴 항목을 선택하여 특정 영상 처리 기능을 수행합니다.
- 선택된 기능에 따라 원본 이미지가 처리되고 결과가 화면에 표시됩니다.
- 사용자가 '9'를 입력하면 프로그램이 종료됩니다.

# 핵심 부분 요약

## 공통 함수부

- `mallocOriImage()`, `mallocTarImage()`: 원본 및 타겟 이미지에 대한 동적 메모리 할당.
- `freeOriImage()`, `freeTarImage()`: 원본 및 타겟 이미지에 대한 메모리 해제.
- `loadImage()`: 파일에서 원본 이미지를 불러와서 메모리에 저장.
- `saveImage()`: 타겟 이미지를 파일에 저장.
- `displayImage()`: 타겟 이미지를 화면에 출력.
- `printMenu()`: 사용자가 선택할 수 있는 메뉴 항목을 콘솔에 출력.

# 핵심 부분 요약

**영상 처리 함수부**

- `equalImage()`: 원본 이미지를 그대로 타겟 이미지로 복사.
- `addImage()`: 밝기 조절. 사용자가 입력한 값을 원본 이미지의 각 픽셀에 더함.
- `reverseImage()`: 반전. 원본 이미지의 각 픽셀 값을 반전시켜 타겟 이미지에 저장.
- `zoomOut()`: 축소. 지정한 배율로 이미지를 축소.
- `zoomIn()`: 확대. 선형 보간법을 사용하여 이미지를 확대.
- `zoomIn2()`: 확대. 픽셀 복제를 사용하여 이미지를 확대.
- `rotate()`, `rotate2()`: 입력 각도로 이미지를 회전.
- `histoStretch()`: 히스토그램 스트레칭을 통해 명암비를 조정.
- `endIn()`: 엔드-인 탐색을 통해 명암비를 조정.
- `histoEqual()`: 히스토그램 평활화를 통해 이미지의 명암을 균등하게 조정.
- `emboss()`: 엠보싱 필터를 적용하여 이미지를 양각 효과로 변환.
- `blurr()`: 블러링 필터를 적용하여 이미지를 흐리게 만듦.
- `edge1()`, `edge2()`, `edge3()`: 서로 다른 마스크를 사용하여 엣지 검출.
- `sobelEdgeDetection()`: Sobel 마스크를 사용하여 수평 및 수직 경계선을 검출.
- `robertsEdgeDetection()`: Roberts 마스크를 사용하여 경계선을 검출.
- `prewittEdgeDetection()`: Prewitt 마스크를 사용하여 경계선을 검출.

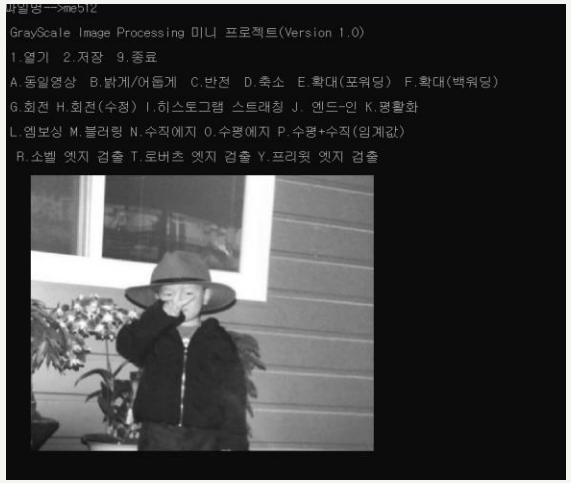
# 각 엣지 검출 함수 요약

- **Sobel 엣지 검출 (`sobelEdgeDetection()`):**
  Sobel 마스크를 사용하여 이미지의 수평(Gx) 및 수직(Gy) 경계선을 검출.
  Gx와 Gy의 결과를 합성하여 최종 경계선을 계산.
- **Roberts 엣지 검출 (`robertsEdgeDetection()`):**
  Roberts 마스크를 사용하여 이미지의 대각선 경계선을 검출.
  Gx와 Gy의 결과를 합성하여 최종 경계선을 계산.

**Prewitt 엣지 검출 (`prewittEdgeDetection()`):**
Prewitt 마스크를 사용하여 이미지의 수평(Gx) 및 수직(Gy) 경계선을 검출.
Gx와 Gy의 결과를 합성하여 최종 경계선을 계산.

# 출력

원본 (me512.raw)




초기화면 (me512.raw)

A. 동일영상

B. 밝게 (값 50)

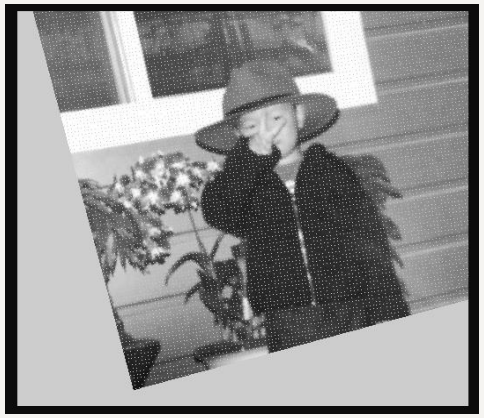
B. 어둡게 (값 -50)

C 반전

# 출력


E. 확대(포워딩) 배율:3배


F. 확대(백워딩) 배율:3배



G. 회전(회전각도 15도)

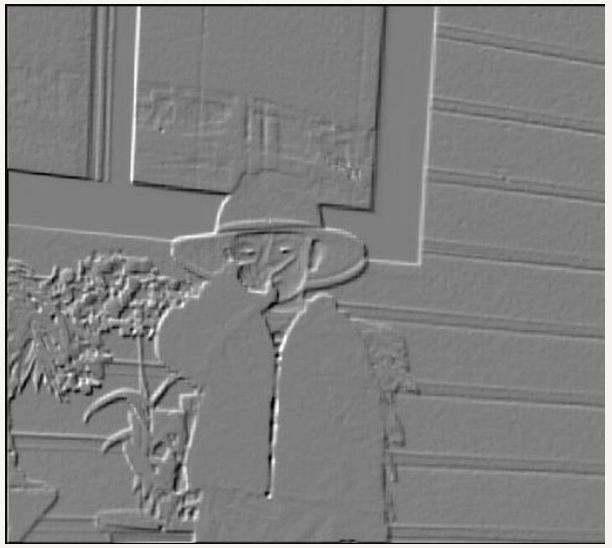
H. 회전수정 (회전각도 15도)

I. 히스토그램 스트레칭

# 출력

J. 엔드-인

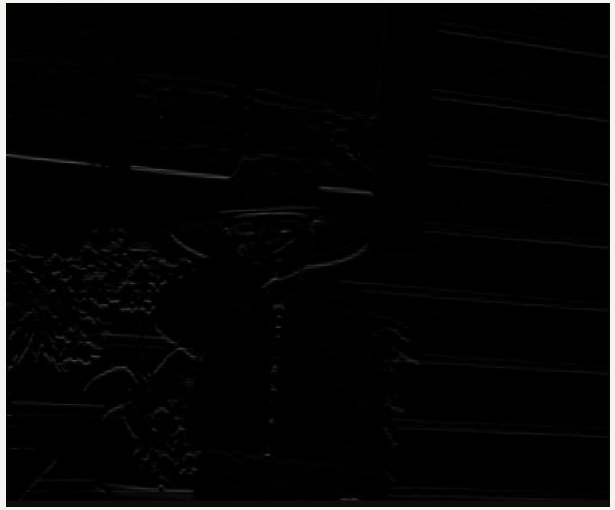
K. 평활화



L. 엠보싱

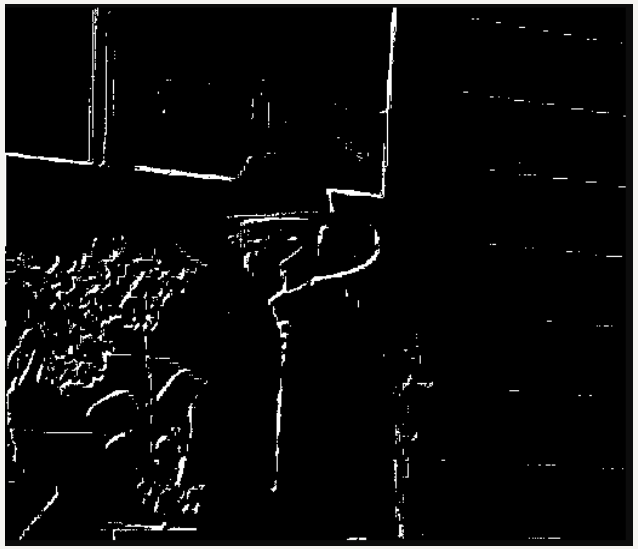
M.블러링

N.수직에지



O.수평에지

# 출력



P. 수평+수직 (임계값 : 20)

R. 소벨 엣지 검출

T. 로버츠 엣지 검출

Y. 프리윗 엣지 검출

# 프로젝트에서 느낀점,한계점,향후 발전 방향

이 프로젝트는 영상 처리의 기초를 학습하고, 실제로 구현해보는 데 큰 도움이 되었습니다.

다양한 한계점이 존재하지만, 향후 발전 방향을 통해 이를 개선하고 확장할 수 있는 많은 가능성이 있는거 같습니다.

더 열심히 배워서 완성도 높은 영상 처리 프로그램으로 만들수 있게 제가 발전할수 있을거 같습니다.