고급 SW 실습 I
Image Filtering
(실습 자료)

CSE4152

서강대학교 컴퓨터공학과

Image Filtering

- 실습 내용
  - (1) Average 필터, (2) Median 필터, (3) Sobel 연산자 등을 시험하는 프로그램 작성
  - OpenCV 함수를 이용해 본다.
  - Average 필터의 경우 추가로 직접 필터를 구현한다.
  - MFC가 아닌 Visual Studio 콘솔 프로그래밍을 사용한다.
- 주의 사항
  - 프로그램 완성 후 담당 조교에게 확인을 받아야 하고 동시에 이들을 사이버 캠퍼스에서 제출하여야 한다.
  - 제출할 파일 이름은 snnnnnnL02_k.cpp로 하여야 한다.
    - 여기서, nnnnnn은 자신의 학번 뒤 6자리. k는 위에 보인 주제 번호.

(**주의**) 만일 파일의 nnnnnn을 자신의 학번 뒤 6자리로 바꾸지 않고, 그냥 snnnnnn_1.cpp 등으로 제출하면 **0점 처리**한다.

주의 사항 (계속)

- 실습 결과 검사
  - 담당 조교가 결과를 검사하면서 제대로 알고 작성했는지 몇가지 작성 내용에 관한 질문을 할 수 있다.
  - 평가 사항이므로 이에 답을 제대로 못하면 감점할 수 있다.
  - 그러니, 프로그램을 작성할 때 내용을 이해하며 작성하여야 한다(질문이 있으면 주저 말고 조교에게 문의할 것)
- 숙제가 있을 경우
  - 제출 파일 이름, 마감일 등을 지정해 줄 것이다.
- 제출 마감
  - 실습 당일 담당 조교가 실습 진행 상황을 감안하여 지정해 줄 것이다. 숙제 역시 마찬가지다.
  - 주의. Late 제출은 절대 허용하지 않는다. 사이버 캠퍼스가 효과적으로 late 제출을 받지 않을 것이다.

Visual Studio Project 생성

- 생성 내용 및 방법
  - VS 콘솔 프로그램을 위한 프로젝트
  - VS2017 실행(1)
    - 파일 → 새로 만들기 → 프로젝트 → Visual C++ → 일반 → 빈 프로젝트 선택
    - 프로젝트 이름 입력(예: swLab19f) → 폴더 선택 → 확인
  - OpenCV 연결
    - 1 주차 MFC 실습 때와 동일.
    - 1 주차 실습 자료를 참고하여 x86 모드에서 설정한다(2).
    - 디버그 모드와 릴리즈 모두 설정한다(실습 때 필요하다)
    - 메모리 누수를 위한 설정도 할 것(1주차 실습 자료)

(1) VS2015, VS2019도 사용 가능할 것이다.
(2) X64에서 작업을 원한다면 이는 개인적으로 해보자. 이 경우 x86과 동일한 설정 작업을 해야 한다(라이브러리에 d가 붙어있지 않다).

실습 전 알아야 할 사항

- 프로그램 작성 스타일
  - 모든 내용을 하나의 파일 snnnnnn.cpp에 작성한다.
  - 이는 검사를 위함이다(실제로는 이렇게 하면 안된다).
- OpenCV 함수
  - OpenCV 함수는 너무나 많아, 이를 다 아는 것은 불가능하다.
  - 다음 document 사이트를 이용하자
    - 2.4.13.6 : https://docs.opencv.org/2.4.13.6/
    - 4.1.1 : https://docs.opencv.org/4.1.1/
      (여기가 더 잘나와 있다. 여기서, 찾은 후 사용법은 2.4.13.6에서 얻는 것도 방법이다. 버전마다 사용법이 약간씩 다르다)
    - 혹은, 원하는 주제를 OpenCV 키워드와 함께 인터넷 검색을 하면 대부분 검색된다.

OpenCV 함수들

- imread( ) (<opencv2/highgui.hpp>)
  - 이미지를 읽어 Mat 에 저장하는 함수.
  - 함수 정의

```
Mat imread(const string& filename, int flags=1 );
                          // flags = 1은 default 값
```

  - 이 함수로 거의 모든 종류의 이미지를 읽을 수 있다. (.bmp, .jpeg, .png 등)
  - flag : 심볼을 사용하면 편리하다. 아래 일부를 보인다.

```
CV_LOAD_IMAGE_COLOR      // color로 읽는다(gray는 gray)
CV_LOAD_IMAGE_GRAYSCALE // grayscale로 읽는다(color도)
```

createTrackbar (<opencv2/highgui.hpp>)

- 이미지 출력 창에 파라미터 값을 조정할 수 있는 슬라이더 추가 기능.
- 콜백 함수를 추가하여, 파라미터 값을 바꿀 때마다 이를 호출할 수 있다.
- 주의: 슬라이더를 조정할 때 값이 0부터 1씩 순차적으로 증가하므로 0보다 큰 홀수 값이어야하는 커널 크기를 적절히 처리해야 한다(1).
- 함수 정의
  - 다음 쪽에서

(1) 실습시 설명

✦함수 정의

```
int createTrackbar(const string& trackbarname,
  const string& winname,   int* value,   int count,
  TrackbarCallback onChange=0,    void* userdata=0 )
```

- trackbarname : 트랙바 이름
- winname : 트랙바를 추가할 윈도우 이름
- value : 트랙바 시작 위치
- count : 트랙바 최대값 (최소값은 항상 0).
- onChange : 이 함수는 트랙바 위치가 변경될 때마다 호출된다. 이의 prototype은 "**FunctionName(int, void*)**" 이어야 한다. '**int**' 값은 'value' 값을 의미한다. '**void***' 는 사용자가 '**userdata**' 로 함수에 전달하는 포인터 값이다. (다음 인자 참고)
- userdata : 이 포인터 변수는 함수 **FunctionName**의 두번째 인자로 전달된다.
- https://docs.opencv.org/2.4/modules/highgui/doc/user_interface.html?highlight=namedwindow

blur( ) (<opencv2/imgproc.hpp>)

- OpenCV Average 필터
- 함수 정의

```
void blur(InputArray src, OutputArray dst,
          Size ksize, Point anchor = Point(-1,-1),
          int borderType = BORDER_DEFAULT )
```

- src : 입력 이미지
- dst : 필터링 후 이미지. 크기와 타입은 입력 이미지와 동일.
- ksize : 커널 크기. 0보다 큰 홀수 값 사용.
- anchor : 커널의 중심점을 설정. 기본값은 커널의 중심 위치.
- borderType : 이미지 밖 픽셀 값 설정 방법(다음 쪽 참조)
- https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=blur#cv2.blur

✦borderType

속성	의미
cv2.BORDER_CONSTANT	iiiiii | abcdefgh | iiiiiii
cv2.BORDER_REPLICATE	aaaaaa | abcdefgh | hhhhhhh
cv2.BORDER_REFLECT	fedcba | abcdefgh | hgfedcb
cv2.BORDER_WRAP	cdefgh | abcdefgh | abcdefg
cv2.BORDER_REFLECT_101	gfedcb | abcdefgh | gfedcba
cv2.BORDER_REFLECT101	gfedcb | abcdefgh | gfedcba
cv2.BORDER_DEFAULT	gfedcb | abcdefgh | gfedcba
cv2.BORDER_TRANSPARENT	uvwxyz | abcdefgh | ijklmno
cv2.BORDER_ISOLATED	영역 (ROI) 밖은 고려 안함

medianBlur( ) (<opencv2/imgproc.hpp>)

- OpenCV median 필터
- 함수 정의

```
void medianBlur(InputArray src, OutputArray dst,
                int ksize)
```

- src : 입력 이미지
- dst : 필터링 후 이미지. 크기와 타입은 입력 이미지와 동일.
- ksize : 커널 크기. 0보다 큰 홀수 값 사용.
- https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=medianblur#cv2.medianBlur

Sobel() (<opencv2/imgproc.hpp>)

- OpenCV Sobel 필터
- 함수 정의

```
void Sobel(InputArray src, OutputArray dst,
           int ddepth, int dx, int dy, int ksize=3,
           double scale=1, double delta=0,
           int borderType=BORDER_DEFAULT )
```

- src : 입력 이미지.
- dst : 필터링 후 이미지. 크기와 타입은 입력 이미지와 같음.
- ddepth : 출력 이미지의 depth. 기본값 -1(입력과 동일).
  입력 이미지형태에 따라 가능한 ddepth 값은 다음과 같다:
  - src.depth() = CV_8U, ddepth = -1 / CV_16S / CV_32F / CV_64F
  - src.depth() = CV_16U/CV_16S, ddepth = -1 / CV_32F / CV_64F
  - src.depth() = CV_32F, ddepth = -1 / CV_32F / CV_64F
  - src.depth() = CV_64F, ddepth = -1 / CV_64F

다음 쪽에 계속

✦ 함수 정의 (계속)

```
void Sobel(InputArray src, OutputArray dst,
           int ddepth, int dx, int dy, int ksize=3,
           double scale=1, double delta=0,
           int borderType=BORDER_DEFAULT )
```

- dx : x에 대한 미분 차수.
- dy : y에 대한 미분 차수.
- ksize : Sobel 커널 크기. 반드시 1, 3, 5, 7 중 하나이어야 함.
- scale : 계산된 값에 적용할 추가 배율.
- delta : 계산된 값에 추가할 상수 값.
- borderType : 이미지 밖 픽셀 값 설정 방법(blur( )의 그것과 동일)
- https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=sobel#cv.Sobel

실행시간 측정

- 필터링 함수 호출 전과 후에 다음 코드를 추가하여, 실행시간을 측정하고 이를 출력한다.
- 이 측정은 함수 시작 후 마칠 때까지 걸리는 시간을 측정한 것이므로, CPU load에 따라 매번 다를 수 있다(1).

```
#include <time.h>
 ...
clock_t start,end;  // global로 선언 가능
...
start = clock();
  // 함수 호출(또는 많은 시간이 필요한 주요 코드)
end=clock();
  // 둘 중 하나 사용
double time_msec = end - start;      // msec 단위
double time_sec = (end - start)/CLOCK_PER_SEC;
                                     // 초 단위
```

(1) 다른 프로그램과 비교를 위하여 제대로 측정하려면, 컴퓨터 load를 가능한 줄인 다음에 여러 번 측정해서 평균을 구해야 한다.

프로그램 예

- 이진 이미지 만들기
  - 기능
    - 읽은 이미지의 픽셀 값이 주어진 threshold 보다 작으면 0, 크거나 같으면 1로 설정하여 이진 영상을 만든다(1).
  - 이 프로그램을 관찰하여 이를 토대로 이어질 실습 문제를 스스로 프로그래밍해보자.

(1) 출력 이미지를 grayscale로 만들 것이므로 1인 경우 255를 설정.

프로그램 소스 코드

```
#pragma once                    // header 파일은 한번만 컴파일
#include <iostream>
#include <time.h>               // 시간 측정 위해서
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>

using namespace cv;

Mat input_im, dst;
clock_t start_time, end_time;

void binaryThresholding_opencvCommand(int kernelsize,
     void *userData);
string type2str(int type); // Mat type을 문자열로 변환
                  // 하여 반환하는 함수(1주차에서 이미 사용)
```

✦ 프로그램 소스 코드(계속)

```
void binaryThresholding_opencvCommand(
      int threshold_value, void *userData) {
  string &win_name =
                  *(static_cast<string*>(userData));
  start_time = clock();
  // 시간 측정할 함수 호출
  threshold(input_im, dst, threshold_value, 255,
            THRESH_BINARY); // call OpenCV function
  end_time = clock();
  cout << "Threshold value : " << threshold_value
       << ", Exec Time : "
       << (double)(end_time - start_time)
       << " (msec)" << endl;
  imshow(win_name, dst);
}
```

✦ 프로그램 소스 코드(main)

```
int main(int argc, char *argv[]) {
  if (argc != 2) { // 파일 경로 및 이름은 argv[1]에
    cout << "[프로그램 사용법]" << endl;
    cout << "명령문 : ~.exe image_file<ent>" << endl;
    return 0;
  }
  input_im = imread(argv[1], CV_LOAD_IMAGE_GRAYSCALE);
  if (input_im.empty()) {
    cout << "File open Error!" << endl;
    return -1;
  }
  cout << "Image size :" << input_im.size()
       << ", Type:" << type2str(input_im.type())
       << endl;
  string window_name = "Binary Converted Image";
  namedWindow(window_name);
```

✦ 프로그램 소스 코드(main 계속)

```
  // Create track bar to change kernel size
  int start_value = 0;
  int max_value = 255;
  createTrackbar( "ThresholdValue", window_name,
    &start_value,  max_value,
    binaryThresholding_opencvCommand,
    static_cast<void*>(&window_name));

  imshow(window_name, input_im);
  waitKey(0);
  return 0;
}
```

실습

- Average 필터링
  - 주어진 이미지를 average 필터를 사용하여 blurring 한다.
  - Trackbar를 사용하여 kernel 크기를 조정할 수 있게 한다. (3 ~ 15, 홀수).
  - 이미지 밖 픽셀은 0으로 처리한다.
  - 계산 시간을 출력한다.
  - 프로그램 실행은 다음과 같이 하도록 한다:

```
C:\실행파일 이미지 flag
```

```
flag = 1 : OpenCV 함수 사용
       2 : 직접 작성한 함수 사용
```

  - 배포 자료에 실행 파일이 있으니 실행해 보고, 그것과 동일한 입출력이 되도록 작성한다.

Median 필터링

- 주어진 이미지를 median 필터를 사용하여 blurring 한다.
- 프로그램 실행은 다음과 같이 한다(flag 불필요)

```
C:\실행파일 이미지 flag
```

- 나머지 사양은 average 필터의 사양과 동일하다.
- 배포 자료에 실행 파일이 있다. 실행해 보자.

Sobel 필터링

- 주어진 이미지를 Sobel 필터를 사용하여 filtering 한다.
- 프로그램 실행은 다음과 같이 한다(flag 불필요)

```
C:\실행파일 이미지 flag
```

- Kernel 크기 제한에 주의하자.
- 나머지 사양은 average 필터의 사양과 동일하다.
- 배포 자료에 실행 파일이 있다. 실행해 보자.