Project 1: Image Filtering (& Numpy Vectorization)

요약 Summary

- 마감: 2024.03.26(화) 23:59
- Part 1: filtering 을 위한 기본 함수 programing
  - 파일 `student.py` 내 함수 **`my_imfilter()`** 본문을 `2d correlation (filter 를 flip 하지 않는 convoltution 연산)` 기능이 수행되도록 구현하세요. 해당 함수는 수업 시간에서 학습한 방식대로 이중 for 문을 사용하는 방식으로 구현하시오.
  - **Mean** filter, **Gaussian** filter, **Sobel** filter 의 kernel 을 생성하고 **`my_imfilter()`** 함수를 호출하여 `filter` 를 수행하시오.
  - `proj1.ipynb, student.py` 파일의 주석 설명문을 자세히 읽고 작업하세요.
- Part 2: 보고서
  - 각 세부 과정에 대한 관련 이론을 요약하여 구현된 코드와 관련지어 설명하세요.
  - 기본 제공된 `Data` 폴더의 예시 영상들에 대해 도출된 결과를 분석하세요. 허용되는 경우, filter 의 크기를 다양하게 변형해 보면서 결과를 도출하고 분석하세요.
- Part 3: +α (추가 점수)
  - 함수 `my_imfilter()`와 동일한 기능을 수행하지만, NumPy 의 vectorization 기능을 활용하여 for 문의 반복 수행을 최대한 적게 사용하도록 `student.py` 내 함수 **`my_imfilter_vect()`**의 본문을 구현하세요. 두 filtering 함수의 수행속도를 측정하고 비교하시오.
- 제출방법:
  - `student.py` 파일, `proj1.ipynb` 파일, 보고서를 하나의 파일로 압축하여 가상대학에 업로드 하세요.
  - 압축파일 파일 이름은 `hw1_본인학번.zip` 으로 하세요. (ex: `hw1_20201234.py`)
  - `student.py` 파일은 파일 이름을 `student_본인학번.py` 으로 변경하세요. (ex: `student_20201234.py`)
  - 보고서 파일는 report_본인학번.pdf 로 지정하세요. (ex: `report_20201234.pdf`)

개요 Overview

본 과제에서는 다양한 linear filter 를 적용할 수 있게 하는 convolution (엄밀하게는 correlation) 함수를 구현하고, 다양한 filter 의 kernel 함수를 생성하여 filtering 결과를 도출합니다.

시작하기 Getting Started

학생 여러분이 직접 구현할 코드는 모두 `student.py` 파일에 구현하면 됩니다. 우선 `my_imfilter()` 함수를 구현한 후, `proj1.ipynb` 파일의 각 셀에 해당되는 filter 의 kernel 함수의 배열을 생성하고 `my_imfilter()` 함수를 호출하여 결과를 확인하세요. 이때 해당 파일 내에 설정된 경로, 입력 값, 입력 parameter 등은 원하는대로 자유롭게 변경해 가며 다양한 결과를 도출해 보는 것을 권장합니다.

아래의 평가기준도 자세히 확인하고, 함수 내의 주석으로 표기된 설명문도 자세히 읽어서 확인하세요. 영문으로 된 설명은 직접 해석을 하기 어려우면 Papago 등 번역 사이트를 활용하세요. 아직 Python 프로그래밍이 익숙하지 않은 학생은 제공되는 Python 프로그래밍 tutorial 문서들을 학습하고, 특히 Numpy tutorial 을 확인하세요. 그리고도 모르는 부분은 가상대학 질의응답 게시판에 편하게 질문을 올려주세요.

요구사항 Requirements / 평가기준 Rubric

- 0 점: 구현한 함수가 제대로 동작하지 않거나, 결과를 아예 도출하지 못하거나 수행 중에 프로그램이 멈추는 경우.
- +50 pts: `student.py` 파일 내 `my_imfilter()`를 적절하게 구현함. 구현한 알고리즘은 다음 조건을 충족해야 함. 아래 조건 중 충족되지 못하는 조건 별로 감점이 될 수 있음.
    i. 영상의 가장자리에서 kernel 이 벗어나지 않도록 input image 의 크기를 kernel 의 크기에 맞춰서 적절하게 늘려주고, 해당 부분의 값에는 zero 값을 채워주는 zero-padding 을 수행함.
    ii. 입력 영상이 흑백 영상 및 컬러 영상일 경우 각각에 맞춰서 적절하게 filtering 을 수행함.
    iii. Filter kernel 의 크기는 자유롭게 홀수 크기 사이즈를 가질 수 있도록 함 (e.g., 7x9 filter 적용 가능해야 하지만 4x5 filter 입력은 대응하지 않아도 됨).
    iv. 혹시 filter kernel 에 짝수 크기를 갖는 입력이 들어올 경우 함수 수행이 되지 않고 에러를 발생시키도록 함
    v. identity filter 에 대해서는 input image 와 동일한 결과가 반환되어야 함.
    vi. 도출된 결과 영상은 input image 동일한 해상도를 가져야 함.

- +30 pts: `proj1.ipynb` 파일 내 각 셀의 설명에 따라 filter 별 kernel 함수의 배열을 적절하게 생성하는 코드를 구현함. 필터 별로 총점/함수개수의 배점을 가짐.

- +20 : 구현한 프로그램에 대해 적절하게 보고서를 작성함.
    - 최소 결과 영상을 3 개씩 제시하세요. (`data` 폴더에 기본 실험용 영상 샘플 제공됨.) 관련되어 연관성이 있는 정보나 설명을 적절하게 제시하세요.
    - MS word, 아래아 한글 등의 문서 편집 프로그램을 사용하여 보고서를 작성하되, 제출 파일은 pdf 형식으로 제출하시오.
- -5 점 : 위 설명 사항을 제대로 따르지 않은 건 별로 감점함.

+α (추가 점수):

- +10 pts: NumPy 의 vectorization 을 이용하여 for 문의 반복 수행을 최대한 적게 사용하는 함수 **`my_imfilter_vect()`**를 구현하고 **`my_imfilter()`** 함수와 수행 시간를 비교하여 시간 단축을 확인함.

사용 불가 함수: 외부 filter, convolution, correlation 함수 일체. 예를 들어 `numpy.convolve()`, `scipy.signal.convolve2d()`, `scipy.ndimage.convolve()`, `scipy.ndimage.correlate()` 등.
사용 가능하고, 유용할 수 있는 함수: 기본적인 numpy 연산자. 덧셈 `numpy.add()` (또는 + 연산자), 원소별 곱셈함수 `numpy.multiply()` (또는 * 연산자), 시그마연산

연산자 `numpy.sum()`, flipping 함수 `numpy.flip`, 범위제한 함수 range
clipping `numpy.clip()`, padding 함수 `numpy.pad()`, 회전 함수 `numpy.rot90()`, 등등.