**Image Processing & Vision Homework 1: Image Filtering**

예술공학대학 컴퓨터예술학부

20190807 민정우

**Implement 2D Gaussian Filter Function in Python**

|  |  |
| --- | --- |
|  | 2차원 가우시안 함수  를 적용하기 위해 배열의 인덱스 를 로 변환하고 이중 반복문을 통해 배열의 각 요소에 대해 가우시안 함수를 적용한다.  필터의 모든 요소의 합이 1이 되지 않으면 이미지가 밝아지거나 어두워지기 때문에 정규화를 통해 필터의 모든 요소의 합이 1이 되도록 한다. |

2D 가우시안 필터 함수 최적화 1

이전 코드는 반복문을 순회하면서 이미 계산한 값을 다시 계산하기 때문에 효율성이 떨어진다.

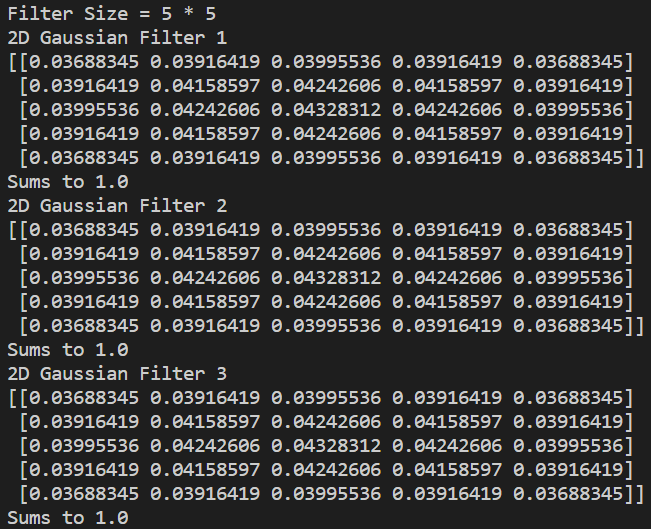
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | |  |  |  | | --- | --- | --- | | (-1, 1) | (0, 1) | (1, 1) | | (-1, 0) | (0, 0) | (1, 0) | | (-1,-1) | (0,-1) | (1,-1) |   필터의 중앙으로부터 같은 거리의 요소들은 중복된 값을 갖는다. 따라서 특정 거리에 대한 가우시안 함수 값을 미리 계산해 저장하면 중복된 계산을 방지할 수 있다.  배열의 중앙 인덱스가 일 때, 와 는 각각 부터 까지의 값을 갖고 따라서 는 부터 까지의 값을 갖는다.  가 부터 일 때의 가우시안 함수 값을 gaussian 배열에 저장한다. 그러면 필터의 모든 인덱스를 미리 구해 놓은 가우시안 함수 값으로 적용할 수 있다. |

2D 가우시안 필터 함수 최적화 2

Numpy는 다차원 배열 계산에 최적화되었기에 필터를 제작하는 데 이중 반복문으로 배열의 각 요소를 하나씩 계산하는 것보다 numpy 배열 단위로 계산하는 것이 코드가 간결해지면서도 성능이 향상된다.

|  |  |
| --- | --- |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 필터 배열을 저장할 output\_filter를 선언할 때 각 요소가 인 2차원 배열 대신 각 요소가 인 2차원 배열로 초기화한다.  이후 배열의 인덱스에 개별로 접근하지 않고 배열 전체에 가우시안 함수 값을 바로 적용하면 이중 반복문을 제거하고 numpy의 성능을 최대한 활용할 수 있다. |

텍스트이(가) 표시된 사진

자동 생성된 설명

세 개의 코드 모두 같은 매개변수를 받을 때 동일한 가우시안 필터를 반환하며(좌), 최적화를 거치면서 필터 제작 속도가 확연히 빨라진 것을 확인할 수 있다(우). 반면 전체 필터링 시간은 오차 범위 내에서 동일한 성능을 보였는데 필터를 생성하는 시간은 줄었지만 필터 생성 시간이 이미지 필터링 시간에 비해 매우 작기 때문에 전체 시간을 줄이는 데 유의미한 영향을 주지 못했다.

**Investigate the visual results according to different filter sizes and standard deviations**

필터의 크기가 같을 때, 표준 편차가 증가하면 값이 더 많이 분산되므로 흐림 효과가 커진다.

|  |  |  |  |
| --- | --- | --- | --- |
| image |  |  |  |
| fsize | 15 | 15 | 15 |
| sigma | 1 | 3 | 5 |

표준 편차가 같을 때, 필터의 크기가 증가하면 값이 더 많이 분산되므로 흐림 효과가 커진다. 그러나 거리가 이상인 픽셀은 가우시안 함수의 값이 0에 가깝기 때문에 필터링에 영향을 미치지 못한다. 따라서 표준 편차가 같을 때, 중심부터의 최대거리가 를 넘어서게 되면 필터 크기와 상관없이 같은 정도의 흐림 효과를 가진다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| img |  |  |  |  |  |
| fsize | 11 | 21 | 31 | 41 | 51 |
| sig | 10 | 10 | 10 | 10 | 10 |

필터의 크기가 증가하면 Boundary Effects에 의해 손실되는 정보가 많아진다. 따라서, 필터의 크기와 표준 편차는 중심부터의 최대거리가 를 넘어서지 않는 정도로 설정하는 것이 이상적이다.

|  |  |  |  |
| --- | --- | --- | --- |
| image |  |  |  |
| fsize | 7 | 13 | 19 |
| sigma | 2 | 4 | 6 |

**Implement 1D Gaussian Filter Function in Python and Compare the visual results**

|  |  |
| --- | --- |
|  | 1차원 가우시안 함수  를 적용하기 위해 배열의 인덱스 를 로 변환하고 반복문을 통해 배열의 각 요소에 대해 가우시안 함수를 적용한다.  필터의 모든 요소의 합이 1이 되지 않으면 이미지가 밝아지거나 어두워지기 때문에 정규화를 통해 필터의 모든 요소의 합이 1이 되도록 한다. |

1차원 가우시안 필터 함수도 위의 2차원 가우시안 필터 함수와 같이 값을 미리 저장한 후 numpy 배열 단위로 계산할 수 있으나, 1차원 배열을 탐색하는 시간보다 배열을 생성하는 시간이 더 크기 때문에 오히려 속도가 느려진다.

**Compare the visual results and computational time**

텍스트이(가) 표시된 사진

자동 생성된 설명 2차원 가우시안 함수는 두 개의 1차원 가우시안 함수의 곱으로 표현할 수 있다. 따라서 각 축에 대한 1차원 가우시안 함수를 연달아 적용하면 2차원 가우시안 함수와 같은 결과를 얻을 수 있다.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Original | Filtering with 2D Gaussian | Filtering with 1D Gaussian |

이미지의 크기가 , 2차원 필터의 크기가 일 때, 2차원 가우시안 필터를 통해 필터링하는 경우 이미지의 각 픽셀마다 2차원 필터를 전부 순회하므로 시간복잡도는 인 반면, 1차원 가우시안 필터를 통해 필터링하는 경우 1차원 필터를 순회하는 과정을 두 번 반복하므로 시간복잡도는 이다. 즉, 필터의 크기가 클수록 2차원 가우시안 함수를 통해 필터링하는 것보다 1차원 가우시안 함수를 연달아 적용해 필터링하는 것이 더 빠르다.

텍스트, 창문, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Numpy가 배열 단위 계산을 효과적으로 처리하기 때문에 필터의 크기가 작다면 2차원 배열 단위로 계산하는 시간보다 이미지를 두 번 순회하는 시간이 더 커져 역효과가 발생할 수 있다.

필터링하는 과정에서 손실되는 가장자리를 버렸기 때문에 계산되는 이미지의 크기가 로 줄었고, 따라서 두 경우의 시간복잡도가 각각 , 로 필터의 크기가 커질수록 계산하는 이미지의 영역이 작아지기 때문에 1차원 가우시안 함수를 통한 필터링의 경우 필터의 크기가 커지면서 시간이 단축됐다.