**고급소프트웨어실습1 2주차 과제**

20171672 이정원

2020.9.20

1) Average 필터링을 효율적으로 구현할 수 있는 Moving Average에 대하여 공식을 이용하여 설명한다.

- Average 필터링은 입력 이미지와 커널의 Convolution 연산 결과를 출력 이미지에 할당하는 과정을 거치는데, 이 과정에서 굉장히 많은 중복된 Convolution연산을 수행한다. Convolution 연산 공식은 다음과 같다.

(m\_imagebits 는 입력 이미지 데이터 변수)

Moving Average필터링은 입력 이미지의 첫 번째 Column의 각 Row에 대하여 Convolution 연산을 수행하여 출력 이미지에 할당한 다음, 두 번째 Column부터 직전 Convolution 연산 결과값(mean)에 이전 커널의 가장 낮은 Column 영역 부분만을 그에 대응하는 입력 이미지와의 Convolution 연산을 수행하여 얻은 값(temp1 이라 하자)을 빼 주고, 현재 계산하려는 커널의 가장 높은 Column 영역 부분만을 Convolution 연산을 수행하여 얻은 값(temp2이라 하자)을 더한다. 이를 공식으로 표현하면 다음과 같다.

예를 들어, 현재 계산하려는 Column 영역이 두 번째~네 번째 Column 영역이라고 가정하자. 그렇다면 직전 Convolution 연산 결과값은 첫 번째~세 번째 Column 영역에 대응하는 연산 결과값일 것이다. 그렇다면 직전 결과값에 첫 번째 Column영역 부분과 입력 이미지와의 Convolution 연산 결과값(temp1)을 빼 주고, 네 번째 Column 영역 부분과 입력 이미지와의 Convolution 연산 결과값(temp2)을 더 해 주면 된다. 그 결과 불필요한 연산(두 번째와 세 번째 Column 영역 부분을 다시 계산하는)을 줄일 수 있게 돼 효과적이게 된다.

2) Median 필터링을 효율적으로 구현할 수 있는 방법에 관하여 도식적으로 설명해 보자. 3x3 윈도우 사이즈를 사용하는 경우를 가정하여 구체적으로 설명한다.

- 실습에서의 Median 필터링은 3\*3 윈도우를 이동시켜 가며 범위 밖의 값은 0으로 할당하고, 입력 이미지 픽셀값들을 1차원 배열 medarray에 할당해 매번 sorting 을 해 주어 중간값을 찾아 결과 이미지 픽셀에 할당해 주었다. (QuickMedian 함수 이용) Sorting 작업으로 인해 처리 시간이 오래 걸리는 특징이 있었다.

이를 효율적으로 구현하기 위해 medarray를 매번 새로 만들지 않고, 본래의 배열에 이전 윈도우의 가장 낮은 Column 영역의 픽셀값을 빼 주고, 다음 윈도우의 가장 높은 Column 영역의 픽셀값을 더한다. 그리고 새로 도식적으로 나타내면 다음과 같다. 업데이트된 medarray에 중간값을 찾아 결과 이미지 픽셀에 할당한다. 도식적으로 나타내면 다음과 같다.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | **-** |  |  | **+** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | **-** |  |  | **+** |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

알고리즘은 다음과 같다.

|  |
| --- |
| **for** r = 1 to m\_width **do**  **for** c = 1 to m\_height **do**  **for** k = -w/2 to w/2 **do**  **Remove** 입력 이미지[(r+k)\*m\_width + c - w/2 - 1] **from** medarray  **Add** 입력 이미지[(r+k) \* m\_width + c +w/2] **to** medarray  결과이미지[r\*m\_width + c] = median(medarray) |