[고소실\_2주차과제]2반\_20161595\_배성현

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

Average 필터링의 경우 현재 픽셀에 대하여 커널 크기만큼의 주변 픽셀 값들과의 평균을 구하여 결과 이미지에 할당하게 된다.



커널의 크기가 w\*w라고 할 때, average 필터링의 일반적인 convolution공식은 위와 같은데, 위의 공식을 통하여 한 픽셀의 average값을 구하는데 걸리는 시간은 w\*w라는 것을 알 수 있고, 이미지의 모든 픽셀에 대하여 average값을 구하여야 하기 때문에 이미지의 너비를 width, 이미지의 높이를 height이라고 할 때 총 w\*w\*width\*height만큼의 시간이 걸린다는 것을 알 수 있다. 하지만 이러한 Average 필터링은 굉장히 많은 중복된 연산을 수행한다는 문제가 있고 이는 moving average를 통하여 해결할 수 있다.

Moving average는 첫 칼럼에 해당하는 픽셀들을 제외한 모든 픽셀에서 이전의 픽셀에 대하여 계산한 average 값을 가지고 있다가, average에서 이전 커널 영역에서의 가장 낮은 column 영역과 이에 상응하는 입력이미지와의 convolution값을 빼주고, 이어서 현재 커널 영역에서의 가장 높은 column 영역과 이에 상응하는 입력이미지와의 convolution값을 더해주어 연산하게 되어 중복된 연산을 할 필요가 없어 average 필터링에 비하여 더 효율적이다.

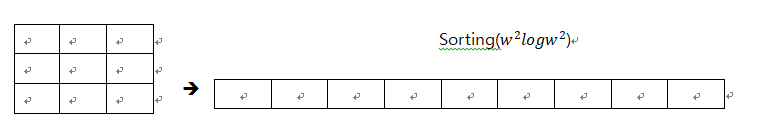
(이전 커널 영역에서의 가장 낮은 column영역에 대한 convolution 값)

(현재 커널 영역에서의 가장 높은 column영역에 대한 convolution 값)

위의 식을 보게 되면 이전 픽셀에 대하여 계산한 average값을 mean이라고 할 때, 현재 픽셀 g(m,n)을 구하기 위해서는 g(m,n)=mean-L-R로 생각할 수 있다. 따라서 moving average를 사용하게 되면 L을 구하는데 w의 시간, R을 구하는데 w의 시간이 걸리므로 g(m,n)을 구하는데 2\*w의 시간이 걸리게 되고 모든 픽셀에 대해서 구하게 되면 약 2\*w\*width\*height의 시간이 걸리므로 average 필터링에 비하여 더 효율적이라고 할 수 있다.

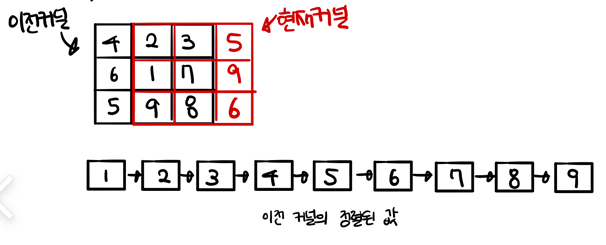
**2. Median 필터링을 효율적으로 구현할 수 있는 방법에 관하여 도식적으로 설명해보자. 3x3 윈도우 사이즈를 사용하는 경우를 가정하여 구체적으로 설명한다.**

Median 필터링은 Average 필터링과 비슷한 방법으로 필터 윈도우를 이동시켜가며, 윈도우 안의 픽셀 값들의 미디언 값을 결과 이미지에 할당하게 된다. Median 필터링은 Sorting 작업을 처리해야 하기 때문에 Average 필터링 등의 convolution 기반 필터링보다 처리 속도가 느리다. 먼저 커널의 사이즈가 w\*w라고 할 때 기존의 Median 필터링의 경우 커널에 대응되는 픽셀들의 미디언 값을 얻기 위해 이를 sorting(Quick Sort)하여야 하기 때문에 만큼이 sorting하는데 들게 된다. 그리고 이를 이미지의 모든 픽셀에 대하여 해야 하기 때문에 총 O( )만큼의 시간이 걸린다고 볼 수 있다.



위의 3\*3커널을 예로 들게 되면 커널의 윈도우 안에 있는 9개의 픽셀 값을 quick sort를 통하여 정렬하기 때문에 총 시간은 height\*width\*9log9의 시간이 들게 된다.

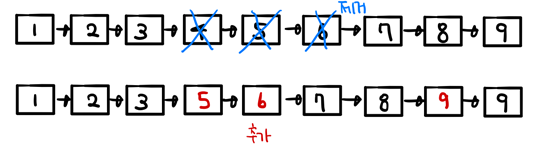
Median 필터링은 linked list를 이용하면 조금 더 효율적으로 구할 수 있는데, 먼저 첫 칼럼에 해당하는 픽셀들을 제외한 모든 픽셀에서 이전 픽셀의 정렬된 값들을 linked list 형태로 저장되어 있게 한다. 그 이후에 해당 linked list에서 이전 커널 영역에서의 가장 낮은 column영역에 해당하는 값들을 제거하여 주고, 이어서 현재 커널 영역에서의 가장 높은 column영역에 해당하는 값들을 추가하여 주면 된다. 이 때 column영역에 해당하는 값들을 제거, 추가하기 전에 미리 정렬하여 주면 더 효율적으로 구현할 수 있다. 3\*3 커널을 사용하는 경우를 가정하여 보면 먼저 아래와 같이 이전 픽셀에 해당하는 커널의 정렬된 값들이 linked list 형태로 저장되어 있다.



그리고 이전 커널 영역에서의 가장 낮은 column 영역에 해당하는 값들을 가져오는데 이 때 3(w)의 시간이 들게 되고, 현재 커널 영역에서의 가장 높은 column영역에 해당하는 값들을 가져오는데 3(w)의 시간이 걸리게 된다. 그리고 이들을 각각 정렬 하게 되면 아래와 같이 각각 3log3(wlogw), 3log3(wlogw)의 시간이 걸리게 된다.



이후 아래와 같이 이전 커널 영역의 가장 낮은 column영역에 해당하는 값들을 제거해주고, 현재 커널 영역에서의 가장 높은 column영역에 해당하는 값들을 순서대로 추가하여 주면 된다.



여기서 제거, 추가 및 median값을 찾는 작업이 약 32(w2)씩 시간이 걸리게 된다. 따라서 한 픽셀의 median값을 구하는 시간 복잡도는 위의 작업들 중 가장 시간이 많이 걸리는 w2인 O(w2)가 되게 된다. 이를 이미지의 모든 픽셀에 대하여 구하게 되면 O(weight\*height\*w2)가 되기 때문에 이와 같이 linked list를 사용하게 되면 기존의 방법인 총 O( )보다 더 효율적이라고 할 수 있다.