**영상처리 n주차 과제**

학번: 20xxxxxxx

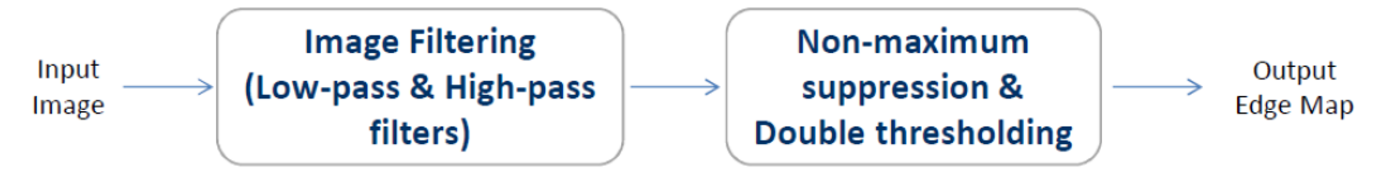
이름: XXX

반드시 .pdf 파일로 저장하여 제출해주세요

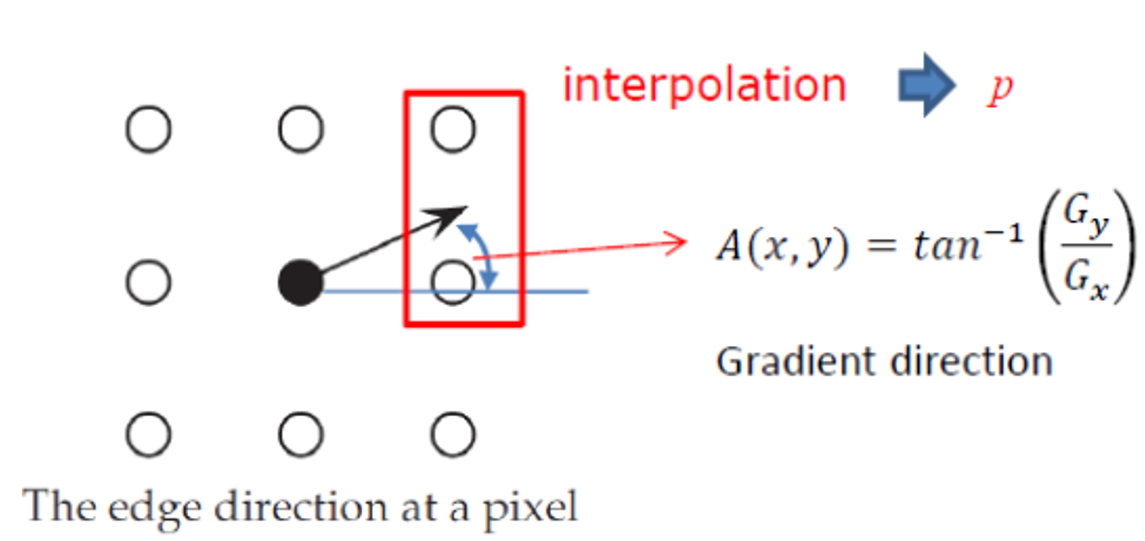
1. **과제 내용**

**:: 구현한 과제에 대한 설명, 어떤 방식으로 접근해야 하는지**

목표 : CannyEdge를 구현하는 것이다



진행과정

1. 입력 이미지가 있으면 , blur를 줘서 노이즈 제거를 한다.
2. Sobel 필터를 통과시켜서 v\_img와 h\_img를 구한 후 대략적인 edge를 찿는다.
3. 찿은 edge 중 진짜 edge인 것이 있고 아닌 것이 있는데, 이중에서 진짜 edge를 찿는 과정을 해야하는데, 이를 위해서, 영상이 가진 magnitude 값과 그 magnitute가 가지는 기울기를 구한다. 그렇게 구한 값으로 주변 픽셀들과의 크기를 비교하여 주변보다 값이 크면 edge이므로 남기고 그렇지않으면 가짜 edge이므로 0으로 지운다.  
   여기서 전부 다 비교할 필요는 없고 벡터의 방향이 가르키는 값과 반대의 값을 비교해준다.
4. 그렇게 가짜들을 제거를 한 후, double threshold를 적용 시킨다.  
   low level filter, high level filter를 정해둔다. 여기서 low보다 낮은 것은 제거, high보다 높은것은 남겨둔다.   
   

이후 나머지 low-level과 high-level 사이에 있는 픽셀 값들에 대해 edge일수도 있고 아닐 수도 있는 값들이 있는데, 이들에 대해 정리작업을 해줘야한다. 이 역시 하나의 픽셀을 잡고 주변 값보다 적으면 지우고 크면 살려두는 식으로 해서 마무리 짓는다.

**구현결과:**

|  |  |  |
| --- | --- | --- |
| Filter\_size=3 | Filter\_size=5 | Filter\_size=7 |
|  |  |  |

1. **구현한 방법에 대한 이유**

**:: 구현한 방법에 대한 설명 ( 왜 이렇게 구현했는지 자세히 )**

|  |  |
| --- | --- |
|  | 우선 사용할 파일들입니다. 캐니를 구현할 캐니엣지 파일과 가우시안 필터와 그에 따라 mirror padding을 할것이므로, 이전에 했던 padding파일과 가우시안 파일을 가져왔습니다. |

|  |
| --- |
|  |
| 매개변수로 이미지와 low-level, high-level,filter\_size를 매개변수로 받습니다.  그후 size를 뽑고,  가우시안 블러를 넣습니다. Sigma는 1로두고, filter\_size는 매개변수로 받은 것을 넣어줍니다.  Pad\_size는 filter에 영향을 받는데 보통 필터사이즈의 크기는 3,5,7인데 이때 1,2,3으로 나누기 2의 몫의 값이 되는 것을 확인 할 수 있습니다.  그 후 이전에 과제하면서 썼던 소벨 필터의 필터 크기를 만들어 주고, 가우시안 블러가 들어간 이미지와의 각각 합성곱을 취해줍니다. |

소벨 필터까지의 결과:

|  |  |
| --- | --- |
| **v\_img** | **h\_img** |
|  |  |

|  |  |  |
| --- | --- | --- |
| 각계산 | | |
|  | | M\_img는 해당 픽셀이 가지는 크기를 나타낸 것이고 Angle\_array는 이미지의 역 탄젠트 값을 구해서, 넣습니다. 여기서 값은 실제 10진수 값이 아닌, 라디안 값으로 방향을 알기 위해 10진수로 변환해야 합니다. 그 과정이 arah가 180를 곱하고 pi로 나누는 것 입니다.  나중에 가르키는 방향에 대한 원할한 처리를 하기 위해 음수 각도에 대해 360을 더해줍니다. 어차피 360들 더한 값도 같은 값을 나타내기 때문입니다. |
| **각 방향에 대한 각도의 처리** | | |
|  | | |
| 방향에 따라 처리 방법이 달라지기 때문에 나눴습니다 | | |
| **angle\_dir이 1,3인경우** | | |
|  | 0~45/ 180~225 일 경우  (i,j)기준 (i+1,j+1)와 ,(i+1,j)와 비교하고, 반대의 벡터를 가르키는 경우 (i-1 ,j-1)와 (i ,j-1)와 비교해서 큰 값을 내야합니다.  저기서 linear-interpolation을 수행합니다.  (nearest아님) | |
| 315~360/ 135~180 일 경우  (i,j)기준 i+1,j-1와 ,(i+1,j)와 비교하고, 반대의 벡터를 가르키는 경우 (i-1,j-1)와 (i ,j-1)와 비교해서 큰 값을 내야합니다.  저기서 linear-interpolation을 수행합니다.  (nearest아님) | |
| **angle\_dir이 2,4인경우** | | |
|  | 45~90/ 225~270 일 경우  (i,j)기준 (i+1,j+1)와 ,(i,j+1)와 비교하고, 반대의 벡터를 가르키는 경우 (i-1 ,j-1)와 (i ,j-1)와 비교해서 큰 값을 내야합니다.  저기서 linear-interpolation을 수행합니다.  (nearest아님) | |
| 315~360/ 135~180 일 경우  (i,j)기준 (i-1,j+1)와 ,(i,j+1)와 비교하고, 반대의 벡터를 가르키는 경우 (i+1,j-1)와 (i ,j-1)와 비교해서 큰 값을 내야합니다.  저기서 linear-interpolation을 수행합니다.  (nearest아님) | |
| **angle\_dir 0 => 0,180,360도에 대한 처리 : i,j일 경우 (i+1,j)와 (i-1,j)와 비교해서 둘다보다 크면 넣고 아니면 0**  **angle\_dir 45 => 45, 225 도에 대한 처리 : i,j일 경우 (i+1,j+1)와 (i-1,j-1) 와 비교해서 둘다보다 크면 넣고 아니면 0**  **angle\_dir 90 => 90,270 도에 대한 처리 : i,j일 경우 (i,j+1)와 (i,j-1) 와 비교해서 둘다보다 크면 넣고 아니면 0**  **angle\_dir 135 => 135,315 도에 대한 처리 : i,j일 경우 (i-1,j+1)와 (i+1,j-1) 와 비교해서 둘다보다 크면 넣고 아니면 0** | | |
| **정확히 여기서 한 것은 linear연산을 위한 1차함수 값의 기울기의 방향을 정하는 것 입니다.** | | |

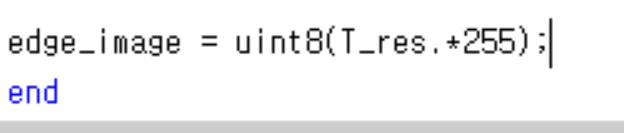
|  |  |  |
| --- | --- | --- |
| Nonmaximum 추출 | | |
|  | | |
| 그리고 기울기를 알아야하므로 다시 라디안으로 각도를 바꿔준 후 각 값에 대한 각도에 대한 탄젠트값을 계산 후 배열로 저장합니다.  먼저 위에서 정의한 대로, 0,45,90,135도에 대한 interpolation을 수행합니다. | | |
| **angle\_dir이 1,3인경우** | | |
|  | | |
| 여기서는 방향이1,3일 때에 대해 수행하는데 오른쪽의 그림과 같은 연산에 대해서입니다.  구해야하는 것은 y값의 범위입니다. 특정y와 특정y+1사이의 s를 구해야 하고 x값은 고정이므로 방정식을 세우자면 기울기는 0.5로 치겠음 (225,330)을 예로 들자면, 330 = 0.5\*225 +b가 되는데  b는 여기서 217.5가 되는 것입니다.  그럼 다음 x+1인 고정이므로 x+1을 대입하면 y+1의 위치의 진y값은  330.5 = 217.5+0.5\*226가 되므로 여기서 s값을 계산하여 interpolation이 가능하고 반대의 경우 여기서 224를 대입하면 계산이 가능합니다 또한 반대라 하더라도 s의 값은 linear한 그래프의 대칭성 때문에 같기 때문에 고려하지않고 그냥 값 계산만 하면 됩니다 |  | |
| **angle\_dir이 2,4인경우** | | |
|  | | |
| 여기서는 방향이2,4일 때에 대해 수행하는데 오른쪽의 그림과 같은 연산에 대해서입니다.  구해야하는 것은 x값의 범위입니다. 특정y와 특정x+1사이의 s를 구해야 하고 y값은 고정이므로 방정식을 세우자면 기울기는 83로 치겠음 (225,330)을 예로 들자면, 330 = 83\*225 +b가 되는데  b는 여기서 -18,345‬가 되는 것입니다.  구해야 하는 것은 x이므로 x를 기준으로한 방정식으로 바꿔줍니다. x =-(y+b)/tan값 = -(330-18345)/83 이식에선 x의 값이 나오므로 y+1인 331을 대입해보면 의 값이 나옵니다. 극단적인 기울기 83의 값에 걸맞는 값이 나오므로 여기서 s를 구한다음에 linear interpolation 연산을 수행하면됩니다.  또한, 반대라 하더라도 s의 값은 linear한 그래프의 대칭성 때문에 같기 때문에 고려하지않고 그냥 값 계산만 하면 됩니다 | |  |
|  | | |

아래는 위의 non-maximum의 연산의 결과 이미지 입니다.

|  |
| --- |
| Non-maximun을 추출한 결과 |
|  |

|  |
| --- |
| Double threshold연산 |
|  |
| If는 로직이 좀 긴데, 현재 픽셀이 low\_level보다 작지도 않고 high레벨보다 크지도 않을 경우 ,즉 그 사이값일 경우 주변 8개의 값 중 하나라도 high-level보다 큰 것이 있다면 그것을 edge로 치겠다는 것입니다. 이어지는 선중에서 연한선이 있을 수 있으므로 같이 없어지는 것을 방지하는 목적임. |

이후 걸러낸 mask에 대하여



를 해준 것이 현재의 결과입니다.

1. **느낀 점**

**:: 구현하면서 느낀 점, 어려웠던 점, 혹은 설명이 필요하다고 느낀 부분**

**수학공부 오랜 만에 제대로 해서 뇌가 시원했습니다. 이전에 해둔 라이브러리들이 있어서 생각보다 어렵지 않게 할 수 있었습니다.**