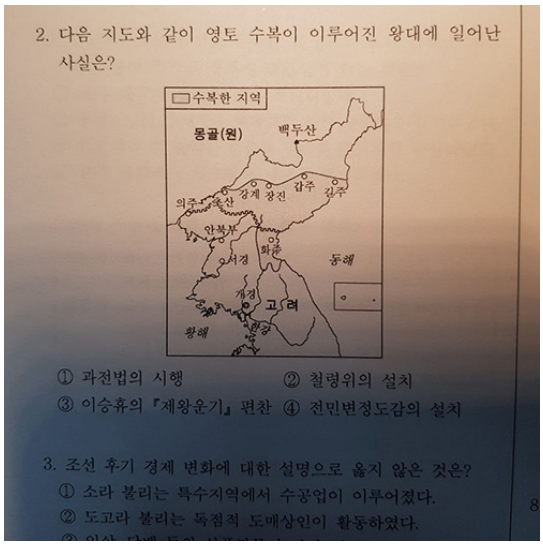
Shadow Remover

1. 문제 상황 분석

주어진 그림들은 프린트 된 문제를 찍은 사진이다. 완벽한 환경에서 사진을 찍었다면 그림자에 대해서 걱정할 필요가 없지만, 만약 책상 위 같은 환경에서 사진을 찍었다면 분명히 조명을 등지거나 조명 바로 위에서 찍게 되어 그림자가 사진에 나타나 있을 것이다. 이번 과제는 이렇게 그림자가 들어간 불완전한 사진(하단)에서 그림자를 제거하여 분명하게 만드는 프로그램을 구현하는 것이다.



여기서 그림자가 있는 그림과 없는 그림에서 공통적으로 나타나는 두 가지 특징을 발견했다. 첫 번째로 사진을 변형시키기 전에는 육안으로 글자를 분명히 읽을 수 있어야 한다는 점이다. 그 이유는 만약 사진에서 그림자가 많이 어두워서 글자와 구분이 안가는 상태라면 의미 있는 데이터를 잃은 상태이므로 어떤 방법을 사용해도 복원할 수가 없다. 마치 Image Warping 에서 모든 점들이 한 점으로(0,0) 이동한 상태라면 다시 원래대로 복원을 할 수 없는 경우와 같다. 두 번째로는 점을 칠할 때는 그 점의 정보만 가져와서는 안 된다는 점이다.

다시 말해서, 원본 사진에서 한 픽셀의 색에 대한 정보를 가져왔을 때, 이 픽셀이 과연 글자인지 아닌지, 즉, 검게 칠 할 건지 하얗게 칠 할 건지를 판단할 때는 그 점만으로는 판단할 수 없다는 이야기이다. 판단할 때는 또 다른 정보가 필요하다는 것을 의미한다. 그래서 나는 첫 번째 특징에서 출력 이미지에서 나는 읽을 수 없는 글자가 존재할 수 도 있다는 것을 인지했다. 그리고 한 점을 찍을 때 그 점 이외의 정보를 가져와야겠다고 생각했다.

2. 문제해결을 위한 아이디어

가장먼저 생각했던 프로그램의 흐름은 크게 다음과 같다. 우선 첫 번째로 이미지가 뿌옇게 찍혀 나와 있을 수 있으므로, 이미지 윤곽을 분명하게 만들어준다. 두 번째로는 선명하게 만든 이미지에서 각 점을 불러와서 찍을지 안 찍을지를 결정하는 과정을 거쳐서 변환된 이미지를 완성한다. 첫 번째 과정에서 Un-sharp Masking 을 사용하는 방법을 사용했었다. 그렇지만 나는 이 문제 상황에 더 적절한 커널을 사용하고 싶었다. 글자 영역의 검출을 한다면 좀더 선명한 이미지를 얻을 수 있을 것이라고 생각했다. 그래서 blur 를 이용한 방법을 사용하지 않고 Cross-correlation filtering(상호-상관 필터)를 사용했다. 커널값을 구상하기 위해서 Laplacian mask(라플라시안 마스크)를 이용하기로 했다.

두 번째 가정은 이 점을 찍을지 안 찍을지에 대한 판단에 대한 것이다. 글자는 분명히 주변 배경보다 어둡다. 그리고 현재 주어진 상황에서는 주변 배경은 종이이다. 종이는 글자를

보여주기 때문에 이미지에서 가장 밝은 영역일 것이다. 그러니까 주변 픽셀들의 평균값이나 중간값을 구한 뒤, 이 값이 종이- 현재 가장 밝은 값이라고 설정을 한다. 이렇게 한다면 그림자가 있어도 그림자영역의 평균이 가장 밝은 부분으로 인식될 것이다. 그리고 여기에 지금 내가 가져온 픽셀이 이 평균(종이)보다 어둡다면 점을 찍으면 된다. 여기서 palette 가 (0,0,0), (주변배경의 평균값)의 색간거리에 대한 Image Threshholding 을 사용하기로 했다.

또, 여기서 주변 배경의 평균값을 계산할 때는 고속으로 처리하기 위하여 지난번 과제 3 : Fastest Mean Filter 에서 만들었던 SAT(Summed Area Table)를 사용했다.

3. 추가적으로 사용한 방법들

처음에는 기본 Laplacian mask 를 사용했었다. 하지만 결과 값이 글자 자체는 상당히 선명하지만 글자 주변부에서 검은 점들이 지저분하게 찍혀있었다. 그래서 마스크를 수정해야 할 필요를 느꼈다. 처음에 사용한 기본 라플라시안 마스크는 (-1, -1, -1, -1, 9, -1, -1, -1, -1) 였다. 최종적으로 내가 적합하다고 생각한 마스크는 (0, -1, 0, -1, 5, -1, 0, -1, 0) 이다. 2 번과 3 번을 비교하면 새로 만든 마스크에서는 글자 주변이 깨끗하다는 것을 확인 할 수 있다.

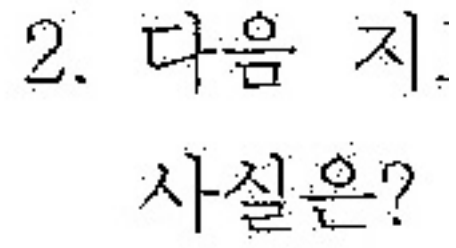


Figure 1: Laplacian

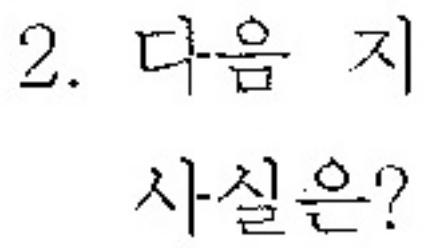


Figure 2: myMask

하지만 이때 기본 라플라시안 마스크를 사용했을 때 발생하지 않았던 문제가 발생했다. 글자가 끊긴 부분이 발생했다. 이렇게 영역 처리를 한 다음에 색을 찍는다. 이때 내가 Image Threshholding 에서 추가한 부분이 있다. 우선 SAT(Summed Area Table)를 이용하여 어느 한 점의 주변 영역의 평균을 구한다.

```
1: for (int i = 0; i < numColor; i++)
2:  {
3:          float d <- diff(f, palette[i]);
4:          if (d < min_d) then
5:          {
6:                  min_color <- i;
7:                  min_d <- d;
8:          }
9:  }
10:  if (min_color == 1) then
11:         palette [1] <- cvScalar(255, 255, 255);
```

end

그리고 그 평균 색을 RGB(255,255,255)대신으로 palette [1]과 가장 밝은 점의 범위로 두고 거리 비교를 진행한다. 반복문에서 가장 먼저 색이 들어가는건 palette[0]인 검은색으로 초기화 될 것이다. 그리고 가장 밝은 색이 들어가는 palette [1]에는 SAT 에서 구한 평균이 들어가게 되는데, 만약에 반복문에서 가장 밝은 색에 가깝다는 판정이 나온다면 이에 따른 처리(10,11 번째 줄)를 해준다. 만약에 판정 결과 평균색과 가깝다는 결과가 나온다면 점은 (255,255,255)의 값이 들어가게 되어 흰 점이 찍히게 될 것이다. 그림자 영역이 있다고 해도 글자는 주변보다 어두울 것이므로 마찬가지로 실행된다. 결과적으로 그림자를 지운 그림이 완성이 된다.