

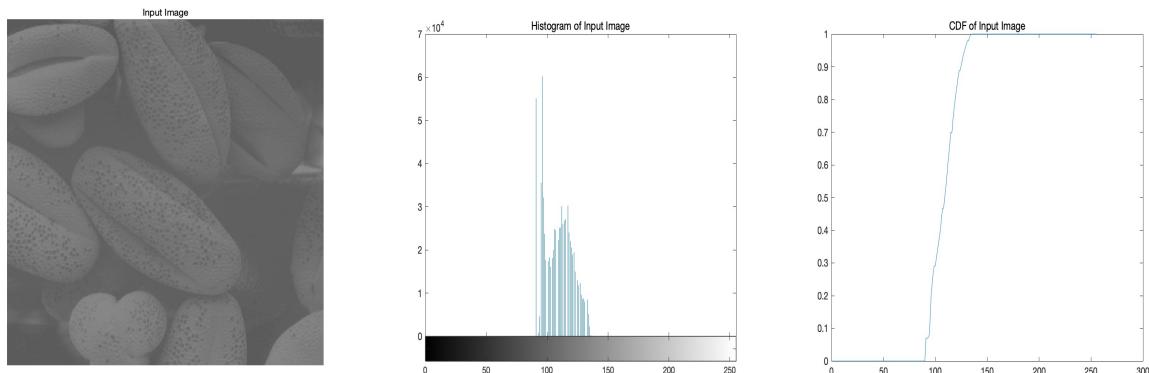
# myCDF

---

## 접근 방법

각 intensity의 분포를 나타내는 행렬을 선언하고 pixel의 개수를 카운팅 하여 행렬의 각 요소를 구한다. 확률을 다 구했다면 변수 하나를 선언하여 누적합을 구해나가면서 output을 구한다. 픽셀의 전체 개수를 구할 때 상수를 대입하기보다는 input으로 주어진 이미지의 차원을 이용하여 구한다. 이렇게 해야지만 **myCDF** 함수를 효율적으로 재활용할 수 있다.

## 결과



## 결과 분석

- CDF는 확률의 누적합이므로 1로 수렴하는 것을 볼 수 있다.
- 과제에서 주어진 이미지의 경우 현재 특정 intensity에 픽셀이 집중되어 있는 것을 볼 수 있다. 즉 intensity의 contrast가 부족하여 이미지의 대비 효과가 적어 눈에 잘 띄지 않는 특징이 있다.

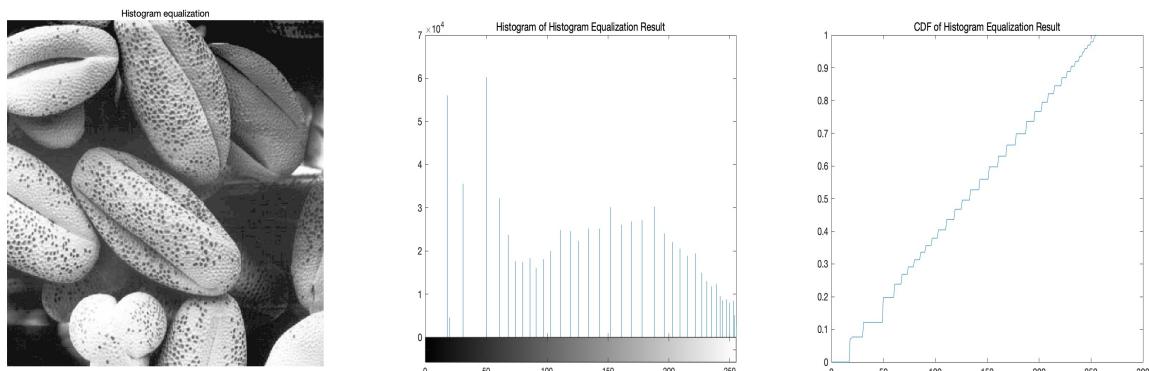
# myHE

---

## 접근 방법

input으로 주어진 이미지의 각 픽셀마다 intensity를 변환하다. 이전에 구한 CDF와 intensity의 범위를 이용하여 변환 이후의 intensity 값을 구한다. skeleton에서는 **myHE** 함수의 input으로 이전에 구한 CDF를 넣어주지 않으므로 다시 한번 **myCDF** 함수를 호출하여 CDF를 구한다.

## 결과



## 결과 분석

- historam이 이전보다 넓게 분포하는 것을 볼 수 있다. 완벽하지는 않지만 중앙 intensity에 집중되어 있던 픽셀이 양옆으로 퍼지게 되었다.
- CDF 또한 완벽하지는 않지만 일정하게 상승하는 모양으로 변한 것을 볼 수 있다.
- 이미지의 contrast가 증가하여 이전보다 이미지의 세세한 부분 등 특징이 더 눈에 잘 띠는 것을 볼 수 있다.

## myAHE

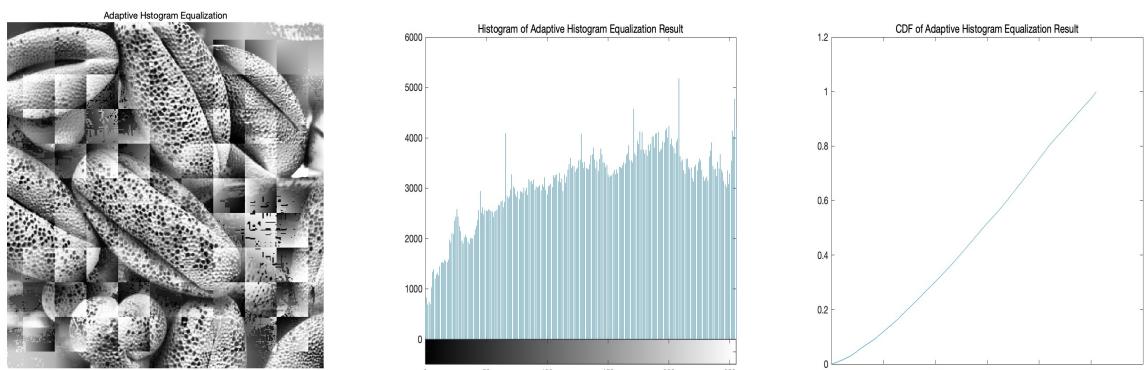
---

### 접근 방법

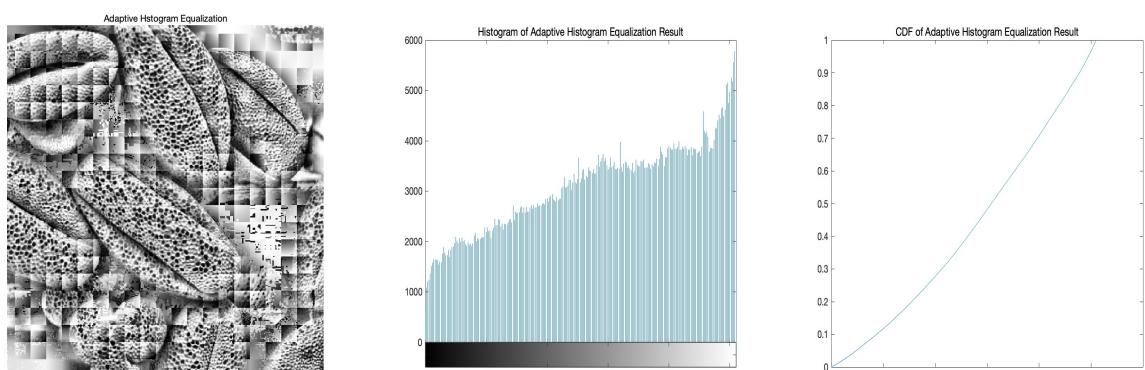
각 tile 별로 CDF를 저장하는 3차원 행렬을 선언한다. 이후 행과 열의 interval을 각각 계산하고 `myCDF` 함수를 이용하여 각 tile 별 CDF를 구한다. input으로 주어진 이미지의 각 픽셀의 intensity를 변경해야 하는데 먼저 특정 픽셀에 가장 근접한 left & top tile의 좌표를 구한다. 이후 시계방향으로 좌표를 회전하면서 유효한 tile의 좌표를 행렬에 저장한다. 위와 같은 과정이 마무리되면 행렬에 저장된 tile의 개수를 기준으로 corner, boundary, center 여부를 판단하고 각 tile의 CDF를 이용하여 intensity 변환 값을 각각 구한다. 최종적으로 boundary라면 linear interpolation, center라면 bilinear interpolation을 진행한다.

### tile 개수 변화에 따른 결과

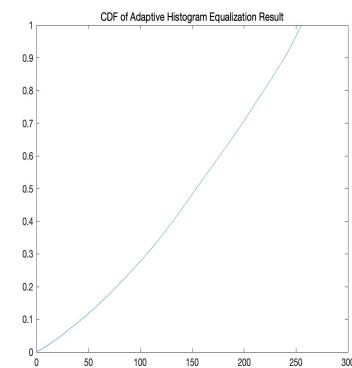
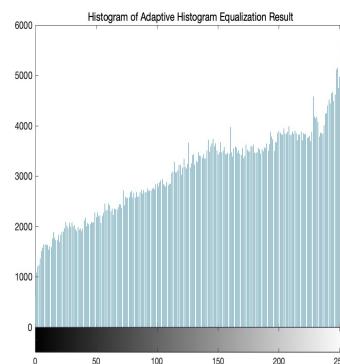
10 x 10



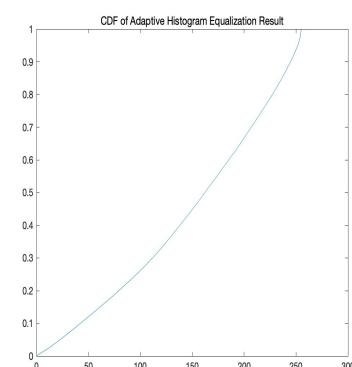
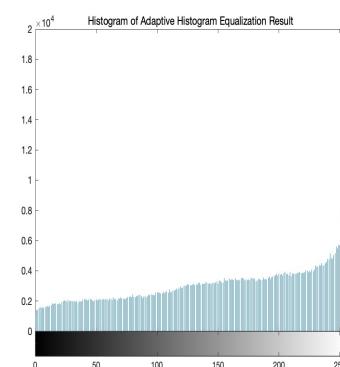
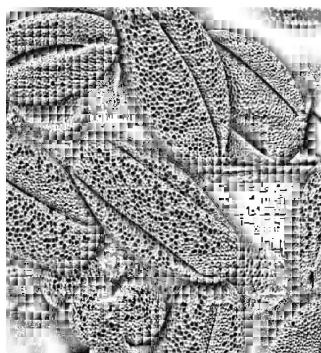
20 x 20



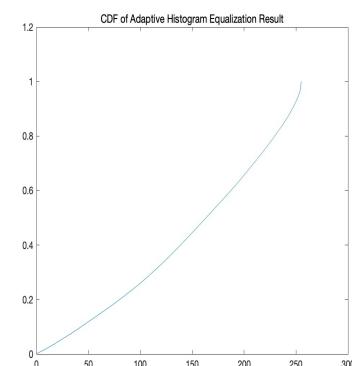
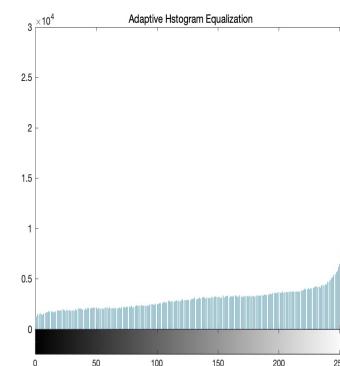
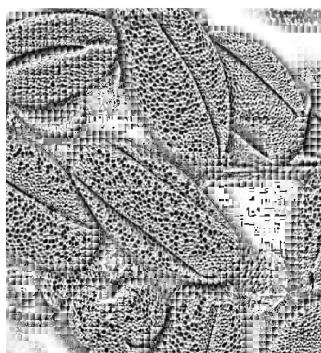
30 x 30



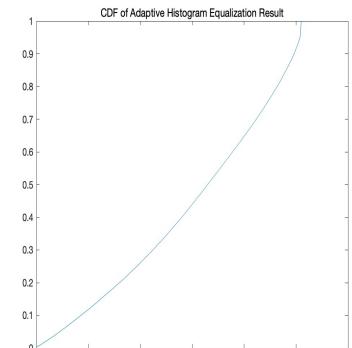
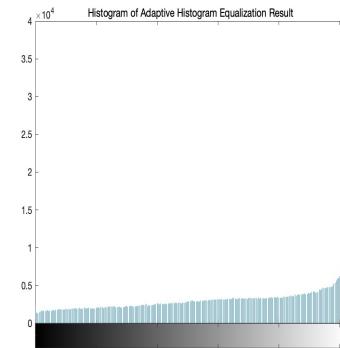
40 x 40



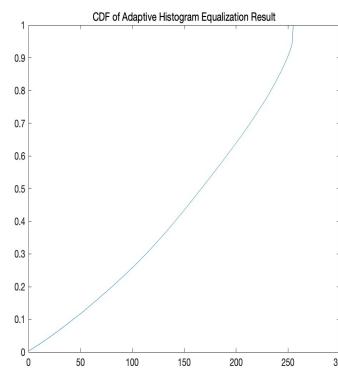
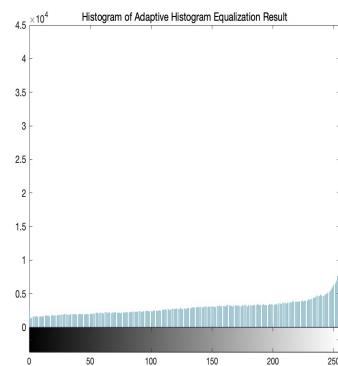
50 x 50



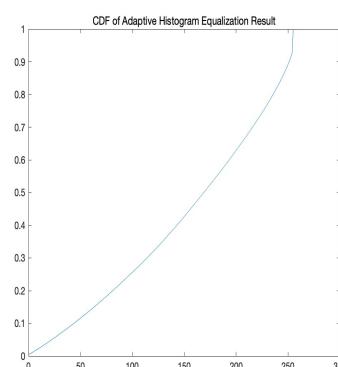
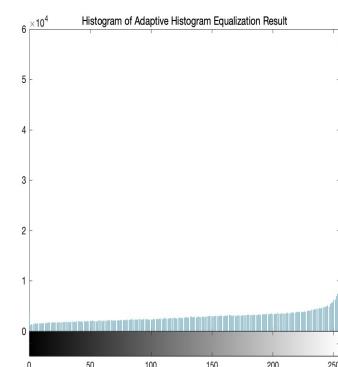
60 x 60



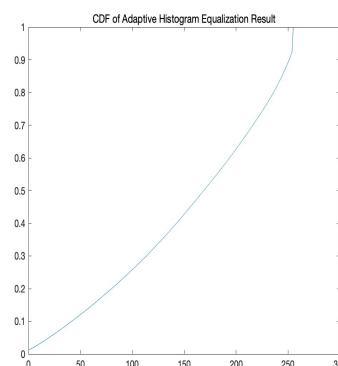
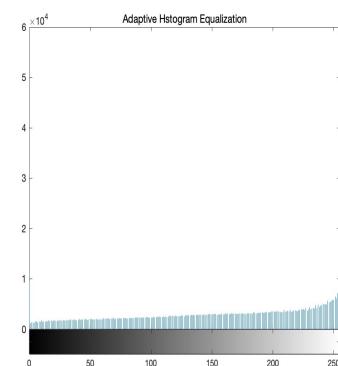
70 x 70



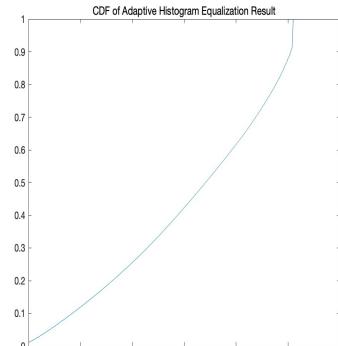
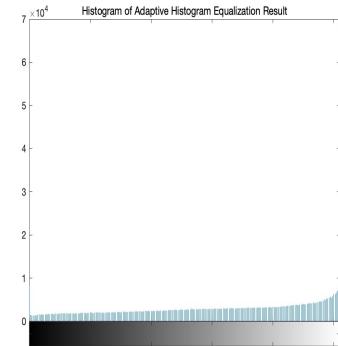
80 x 80



90 x 90



100 x 100



## 결과 분석

- tile의 개수가 증가할수록 histogram이 더 평평해지는 것을 볼 수 있다.
- 특이한 점은 tile의 개수가 일정 범위를 넘어가게 되면 CDF의 마지막 부분에서 pulse처럼 급격히 증가하는 구간이 생긴다는 것이다. tile의 개수가 많아지면 각 tile에 매핑되는 CDF에서 특정 intensity의 확률이 매우 높아지므로 이와 같은 현상이 발생하는 것 같다.
- 위와 같은 이유로 tile 개수가 증가할수록 이미지에서 매우 밝게 존재하는 픽셀의 개수가 점점 증가하는 것을 볼 수 있다.

## 고찰

---

수업 시간에 배운 histogram equalization에 대해 직접 실습해 보면서 깊이 있는 이해를 할 수 있었다. 특히 **myAHE** 함수 구현에 애를 먹었는데 adaptive histogram equalization의 정확한 이해와 체계적인 구현으로 해결할 수 있었다.