전산학특강(영상처리특론)

Assignment 1 Report

고려대학교 컴퓨터학과
정경륜

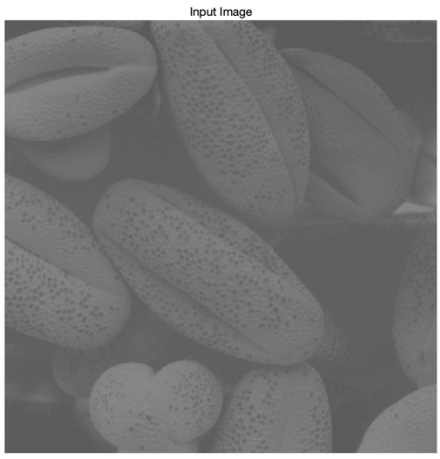
1. Cumulative Distribution Function (myCDF.m)

1) Code

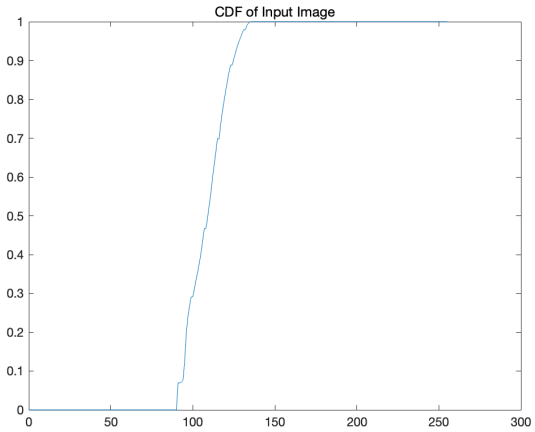
```
1  function output = myCDF(image)
2
3  output=zeros(256,1);
4
5  % todo
6  [M, N] = size(image);
7  MN = M * N;
8  hist=zeros(256,1);
9  for x=1:M
10     for y=1:N
11         hist(image(x, y)+1)= hist(image(x, y)+1)+1;
12     end
13 end
14
15 P=hist(1)/MN;
16 output(1)=P;
17 for i = 2:256
18     P=hist(i)/MN;
19     output(i)=output(i-1)+P;
20 end
21 end
```

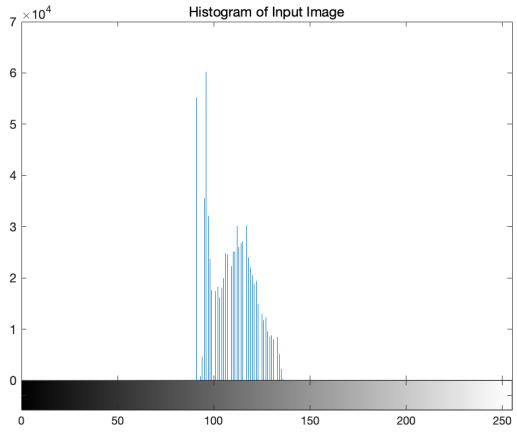
우선 특정 intensity의 확률을 구하기 위해서 이미지의 사이즈와 각 histogram 분포가 필요하다. 6번 라인에서 image의 사이즈를 size()함수를 이용하여 구한다. (size()는 교수님께서 사용해도 된다고 하셨습니다.) 그 후 9번부터 13번 라인을 통해서 histogram 분포를 직접 구한다. 이후 index 1부터 256까지 하나씩 누적합을 구하여 CDF 분포를 구하게 된다.

2) 결과 설명



Input 이미지를 보게 되면 전반적으로 어둡고 constrast가 낮음을 확인할 수 있다.





왼쪽 그림의 CDF를 살펴보면 intensity가 80-90 이전까진 변화가 없다가 그 이후부터 140 정도까지 가파르게 증가하는 모습을 볼 수 있다. 이를 histogram의 분포가 고르지 못 하다는 것을 알 수 있고 오른쪽 그림의 실제 histogram 분포를 살펴봐도 고르게 분포하고 있지 못 하고 특히 intensity가 100 부근에 몰려 있는 모습을 볼 수 있다.

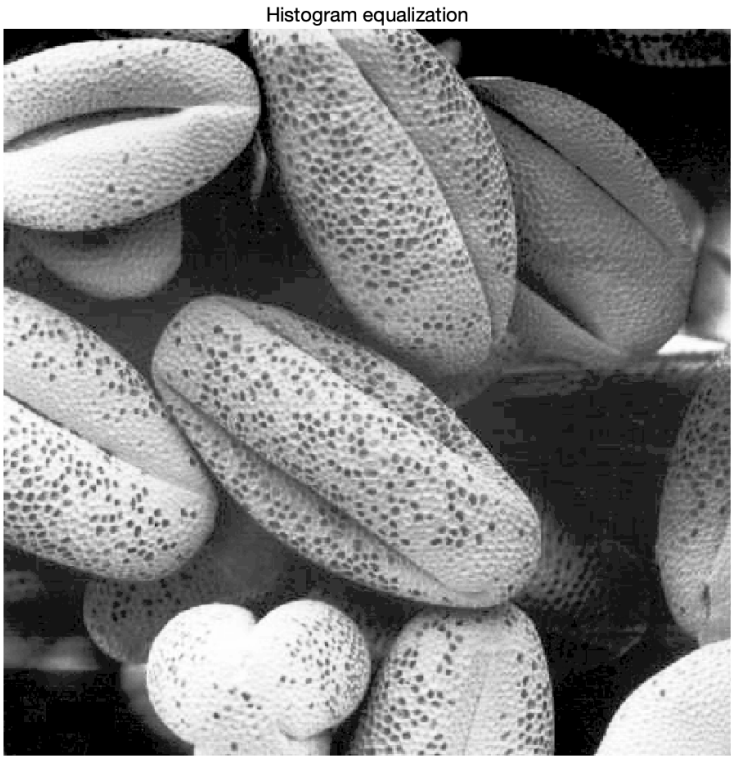
2. Histogram Equalization (myHE.m)

1) Code

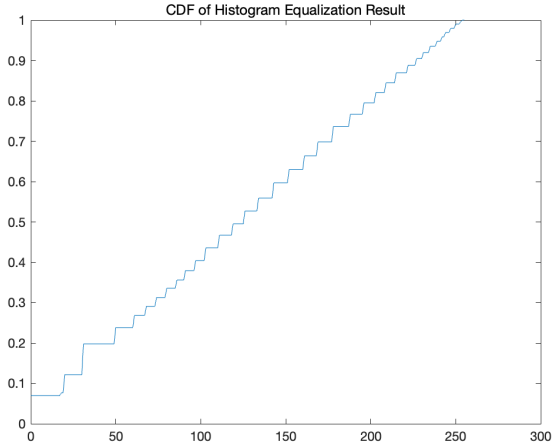
```
1  function output = myHE(input)
2
3  dimX = size(input,1);
4  dimY = size(input,2);
5
6  output = uint8(zeros(dimX,dimY));
7
8  % ToDo
9  L=256;
10 S=zeros(256, 1);
11 cdf=myCDF(input);
12
13 for i = 1:L
14     S(i)=(L-1) * cdf(i);
15 end
16
17 for x = 1:dimX
18     for y = 1:dimY
19         output(x, y)=S(input(x, y) + 1);
20     end
21 end
22 end
```

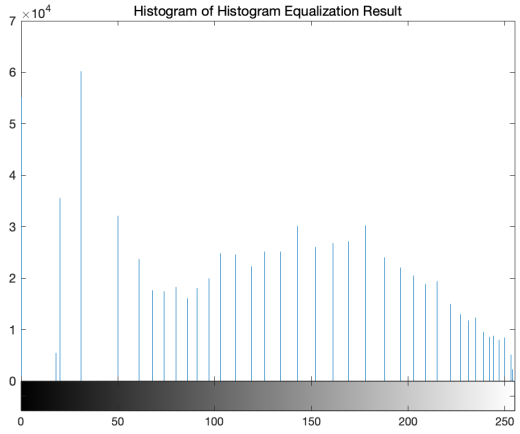
수업 시간에 배운 histogram equalization 수식에 맞춰서 CDF에 (color scale의 수 - 1)인 255를 곱하여 histogram equalization을 적용하기 위한 함수를 만들어낸다. 그 후 17 ~ 21번 라인에서 앞서 구한 함수를 이용하여 histogram equalization이 적용된 intensity를 변환한다.

2) 결과 설명



Histogram equalization의 결과를 보면 앞서 봤던 input image보다 constrast가 분명한 모습을 확인할 수 있다. 또한 input image는 어두운 색감에만 치중되어 있던 반면에 histogram equalization을 거친 이미지는 어두운 부분과 밝은 부분이 동시에 존재하는 것으로 보아 intensity가 전체적으로 고루 분포되어 있음을 시각적으로도 확인할 수 있다.





왼쪽 그림을 보면 histogram equalization을 거친 CDF 를 볼 수 있다. 앞서 봤던 input의 CDF처럼 특정 intensity에서 급격하게 상승하는 것이 아닌 intensity 전체에서 CDF가 선형적으로 증가하는 모습을 볼 수 있다. 물론 CDF 그래프가 계단 형식으로 나타나긴 하나 input의 CDF와 비교 했을 땐 선형적임을 확인할 수 있다. Histogram equalization을 거친 histogram을 살펴보면 이전보다 intensity 전반에 걸쳐서 고루 분포하고 있음을 볼 수 있다. 물론 어두운 쪽에 histogram 값이 좀 더 크긴 하지만 처음 input image보단 고루 분포되어 있다.

3. Adaptive Histogram Equalization (myAHE.m)

1) Code

```
function output = myAHE(input, numtiles)

dimX = size(input,1);
dimY = size(input,2);

output = uint8(zeros(dimX,dimY));

% ToDo
tileX=numtiles(1);
tileY=numtiles(2);
cdfs=zeros(256, tileX, tileY);
S=zeros(256, tileX, tileY);
spaceTileX=ceil(dimX/tileX);
spaceTileY=ceil(dimY/tileY);
spaceLastX=dimX - (tileX-1) * spaceTileX;
spaceLastY=dimY - (tileY-1) * spaceTileY;
```

우선 AHE를 구현하기 위해 필요한 변수들을 정의한다. cdfs는 각 타일 별 CDF 값을 담기 위한 변수이고 S는 각 타일 별 histogram equalization을 위한 변환 함수이다. spaceTile 변수는 한 타일 당 길이이다. spaceLast는 마지막 타일의 길이이다.

```
for i=1:tileX
    for j=1:tileY
        if i==tileX && j==tileY
            cdfs(:,i,j)=myCDF(input((i-1) *spaceTileX + 1 : dimX, (j-1)*spaceTileY + 1 : dimY));
        elseif i==tileX && j~=tileY
            cdfs(:,i,j)=myCDF(input((i-1) *spaceTileX + 1 : dimX, (j-1)*spaceTileY + 1 : j*spaceTileY));
        elseif i~=tileX && j==tileY
            cdfs(:,i,j)=myCDF(input((i-1) *spaceTileX + 1 : i*spaceTileX, (j-1)*spaceTileY + 1 : dimY));
        else
            cdfs(:,i,j)=myCDF(input((i-1) *spaceTileX + 1 : i*spaceTileX, (j-1)*spaceTileY + 1 : j*spaceTileY));
        end
        for k=1:256
            S(k,i,j)=255*cdfs(k,i,j);
        end
    end
end
```

그 후 타일별로 CDF 값을 구하고 이를 cdfs에 저장한다. 마지막 타일의 길이가 input과 타일의 개수에 따라 상이하기 때문에 그 부분만 에러 처리를 따로 진행해 주었다. 이후 29 ~ 30번 라인에서 cdfs를 활용해 histogram equalization function을 저장한다.

```
for x = 1:dimX
    for y = 1:dimY
        tX=floor((x-1)/spaceTileX)+1;
        tY=floor((y-1)/spaceTileY)+1;

        if tX == tileX
            mX=mod((x-1) - (tileX-1) * spaceTileX, spaceLastX);
            spaceX=spaceLastX;
        else
            mX=mod((x-1), spaceTileX);
            spaceX=spaceTileX;
        end

        if tY == tileY
            mY=mod((y-1) - (tileY-1) * spaceTileY, spaceLastY);
            spaceY=spaceLastY;
        else
            mY=mod((y-1), spaceTileY);
            spaceY=spaceTileY;
        end
        mX=mX+1;
        mY=mY+1;
```

이제 35번 라인부터 실제 adaptive histogram equalization을 진행한다. tX, tY는 해당 x, y가 어떤 타일에 속하는 지를 나타내는 변수이다. mX와 mY는 모듈러 연산을 이용하여 각 x랑 y가 속한 타일 안에서의 위치 값을 표현하는 변수이다. 이 또한 마지막 타일일 경우 타일의 길이가 다르기에 따로 계산하여 처리하였다.

```
58          if (tX==1) && (tY==1) && (mX <= (spaceX/2)) && (mY <= (spaceY/2))
59                  output(x, y)=S(input(x, y)+1, tX,tY);
60          elseif (tX==tileX) && (tY==1) && (mX > (spaceX/2)) && (mY <= (spaceY/2))
61                  output(x, y)=S(input(x, y)+1, tX,tY);
62          elseif (tX==1) && (tY==tileY) && (mX <= (spaceX/2)) && (mY > (spaceY/2))
63                  output(x, y)=S(input(x, y)+1, tX,tY);
64          elseif (tX==tileX) && (tY==tileY) && (mX > (spaceX/2)) && (mY > (spaceY/2))
65                  output(x, y)=S(input(x, y)+1, tX,tY);
```

이제 본격적으로 adaptive histogram equalization를 진행한다. 우선 이미지의 모서리 부분을 먼저 처리한다. 이미지의 모서리 타일 중에서도 4등분하여

```
67          elseif ((tY==1) && (mY <= (spaceY/2))) || ((tY==tileY) && (mY > (spaceY/2)))
68              if mX <= (spaceX/2)
69                  leftRatio=(mX + floor(spaceX/2)) / spaceX;
70                  rightRatio=(floor(spaceX/2) - mX) / spaceX;
71                  output(x, y)=rightRatio * S(input(x, y)+1, tX-1, tY) + leftRatio * S(input(x, y)+1, tX, tY);
72              else
73                  leftRatio=(floor(spaceX/2) - (spaceX - mX)) / spaceX;
74                  rightRatio=((spaceX - mX) + floor(spaceX/2)) / spaceX;
75                  output(x, y)=rightRatio * S(input(x, y)+1, tX, tY) + leftRatio * S(input(x, y)+1, tX+1, tY);
76              end
77
78          elseif ((tX==1) && (mX <= (spaceX/2))) || ((tX==tileX) && (mX > (spaceX/2)))
79              if mY <= (spaceY/2)
80                  topRatio=(mY + floor(spaceY/2)) / spaceY;
81                  downRatio=(floor(spaceY/2) - mY) / spaceY;
82                  output(x, y)=downRatio * S(input(x, y)+1, tX, tY-1) + topRatio * S(input(x, y)+1, tX, tY);
83              else
84                  topRatio=(floor(spaceY/2) - (spaceY - mY)) / spaceY;
85                  downRatio=((spaceY - mY) + floor(spaceY/2)) / spaceY;
86                  output(x, y)=downRatio * S(input(x, y)+1, tX, tY) + topRatio * S(input(x, y)+1, tX, tY+1);
87              end
```

```
89          elseif (mX <= (spaceX/2)) && (mY <= (spaceY/2))
90              topRatio = (mY + floor(spaceY/2)) / spaceY;
91              downRatio = (floor(spaceY/2) - mY) / spaceY;
92              leftRatio = (mX + floor(spaceX/2)) / spaceX;
93              rightRatio = (floor(spaceX/2) - mX) / spaceX;
94              output(x, y) = downRatio * ((rightRatio * S(input(x, y)+1, tX-1, tY-1)) + (leftRatio * S(input(x, y)+1, tX, tY-1))) + topRatio * ((rightRatio * S(input(x, y)+1, tX-1, tY)) + (leftRatio * S(input(x, y)+1, tX, tY)));
95
96          elseif (mX > (spaceX/2)) && (mY <= (spaceY/2))
97              topRatio = (mY + floor(spaceY/2)) / spaceY;
98              downRatio = (floor(spaceY/2) - mY) / spaceY;
99              leftRatio = (floor(spaceX/2) - (spaceX - mX)) / spaceX;
100             rightRatio = ((spaceX - mX) + floor(spaceX/2)) / spaceX;
101             output(x, y) = downRatio * ((rightRatio * S(input(x, y)+1, tX, tY-1)) + (leftRatio * S(input(x, y)+1, tX+1, tY-1))) + topRatio * ((rightRatio * S(input(x, y)+1, tX, tY)) + (leftRatio * S(input(x, y)+1, tX+1, tY)));
102
103         elseif (mX <= (spaceX/2)) && (mY > (spaceY/2))
104             topRatio = (floor(spaceY/2) - (spaceY - mY)) / spaceY;
105             downRatio = ((spaceY - mY) + floor(spaceY/2)) / spaceY;
106             leftRatio = (mX + floor(spaceX/2)) / spaceX;
107             rightRatio = (floor(spaceX/2) - mX) / spaceX;
108             output(x, y) = downRatio * ((rightRatio * S(input(x, y)+1, tX-1, tY)) + (leftRatio * S(input(x, y)+1, tX, tY))) + topRatio * ((rightRatio * S(input(x, y)+1, tX-1, tY+1)) + (leftRatio * S(input(x, y)+1, tX, tY+1)));
109
110         else
111             topRatio = (floor(spaceY/2) - (spaceY - mY)) / spaceY;
112             downRatio = ((spaceY - mY) + floor(spaceY/2)) / spaceY;
113             leftRatio = (floor(spaceX/2) - (spaceX - mX)) / spaceX;
114             rightRatio = ((spaceX - mX) + floor(spaceX/2)) / spaceX;
115             output(x, y) = downRatio * ((rightRatio * S(input(x, y)+1, tX, tY)) + (leftRatio * S(input(x, y)+1, tX+1, tY))) + topRatio * ((rightRatio * S(input(x, y)+1, tX, tY+1)) + (leftRatio * S(input(x, y)+1, tX+1, tY+1)));
116         end
```

2) 결과 설명