|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **과목명** | **영상신호처리** | **학기** | **2023 - 1** | **담당교수** | **장주용** |
| **학과** | **전자통신공학과** | **학번** | **2018707072** | **이름** | **이승룡** |
| **과제 1 -** **histogram equalization** | | | | | |

**1, 과제설명**

histogram equalization 알고리즘을 “heq.m”에 구현하라. 코드는 “prob1.m”을 정확 히 풀어야 한다. 즉, “prob1.m” 스크립트가 실행되었을 때 아래와 같은 그림들이 출력되어야 한다. 이때 MATLAB이 제공하는 histogram 관련 built-in function(ex: imhist, histeq, histogram)을 사용하지 않는다.

차트이(가) 표시된 사진

자동 생성된 설명 텍스트, 야외, 하늘, 들판이(가) 표시된 사진

자동 생성된 설명

**2, 이론**

히스토그램 평활화(histogram equalized)는 이미지의 픽셀 강도 값의 분포를 재분배하여, 결과 히스토그램이 전체 값 범위에 걸쳐 균등하게 분포하도록 하는 과정이다. 픽셀 값 분포가 집중된 히스토그램은 대비가 낮은 이미지를 결과로 나타내고, 픽셀 값 분포가 균일한 히스토그램은 대비가 높은 이미지를 나타내기 때문에 히스토그램 평활화는 대비를 향상시키기 위해 사용하는 방법이다.

히스토그램 평활화 과정은 다음과 같다.

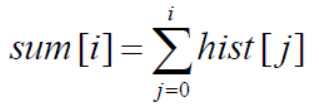
1. 입력 이미지의 히스토그램을 계산.

아래의 히스토그램을 보면 픽셀 값의 분포가 편향되어 있는 것을 확인할 수 있다.

차트이(가) 표시된 사진

자동 생성된 설명

1. 히스토그램의 누적 분포 함수 (CDF) 계산.



차트이(가) 표시된 사진

자동 생성된 설명

1. CDF를 정규화 계산 후 최대 강도 값 (일반적으로 255)으로 곱하여 픽셀의 새로운 강도 값을 계산.

도표이(가) 표시된 사진

자동 생성된 설명

차트이(가) 표시된 사진

자동 생성된 설명

1. 기존의 픽셀 값 대신 새로운 픽셀 값으로 변환.

결과 이미지는 강도 값 범위 전체에 걸쳐 균일하게 분포되는 히스토그램을 갖게 되어 대비가 향상된다.

**3. 소스코드 설명**

|  |
| --- |
| % histogram of input image without imhist function  for i = 1:height  for j = 1:width  %feq(i, j): uint8 -> Can't express 256  pixel\_value = cast(feq(i, j), 'double');  f\_H(pixel\_value + 1) = f\_H(pixel\_value + 1) + 1;  end  end |

입력 이미지의 histogram을 imhist 함수를 사용하지 않고 구하는 코드이다. 위에서 설명한 히스토그램 평활화 과정1에 해당한다. 여기서 cast를 사용한 이유는 feq(i, j)의 type이 uint8이기 때문에 256이상의 값을 표현하지 못하기 때문이다. 여기서 형변환을 하지 않았을 때 생기는 문제점은 고찰에 첨부하였다.

|  |
| --- |
| % Cumulative distribution of H  cH = cumsum(f\_H);  cH = round(cH \* L/cH(end)); % cH(end) = number of pixels |

히스토그램의 누적 분포 함수를 계산하고, 정규화 계산 후 최대 가능한 강도 값을 곱하였다. 위에서 설명한 과정2, 3에 해당한다.

|  |
| --- |
| % output image  for i = 1:height  for j = 1:width  %feq(i, j): uint8 -> Can't express 256  pixel\_value = cast(feq(i, j), 'double');  feq(i,j) = cH(pixel\_value + 1);  end  end |

과정4에 해당하는 기존의 픽셀 값 대신 새로운 픽셀 값으로 변환하는 과정이다.

|  |
| --- |
| % histogram of output image without imhist function  for i = 1:height  for j = 1:width  %feq(i, j): uint8 -> Can't express 256  pixel\_value = cast(feq(i, j), 'double');  Heq(pixel\_value + 1) = Heq(pixel\_value +1) + 1;  end  end |

출력 이미지의 histogram을 imhist 함수를 사용하지 않고 구하는 코드이다.

**4. 실행결과**

텍스트, 야외, 하늘, 흐린이(가) 표시된 사진

자동 생성된 설명 차트이(가) 표시된 사진

자동 생성된 설명

차트이(가) 표시된 사진

자동 생성된 설명 텍스트, 야외, 하늘, 들판이(가) 표시된 사진

자동 생성된 설명

입력 이미지에 비해 히스토그램 평활화 후 이미지의 대비가 향상된 것을 확인할 수 있다. 또한 출력 이미지의 히스토그램의 누적 분포가 일직선 형태를 나타낸다.

**5. 고찰**

처음 코드를 구현했을 때 아래 그림과 같이 X축 값이 255인 부분에서 1보다 큰 값이 나와 이상한 peak값이 결과에서 출력되었다. 그래서 변수들의 값들을 하나씩 확인해보았고 “heq.m”에서 입력 이미지의 히스토그램을 구하는 과정이 잘못된 것을 깨달았다.

차트이(가) 표시된 사진

자동 생성된 설명

peak값이 출력된 결과

테이블이(가) 표시된 사진

자동 생성된 설명

imhist함수를 사용해서 히스토그램 구한 경우

**테이블이(가) 표시된 사진

자동 생성된 설명**

아래의 코드로 히스토그램 구한 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

f\_H의 255인덱스의 값을 보면 H의 255, 256번째 값의 합과 동일한 것을 알 수 있다. 이는 feq(i, j)의 type이 uint8이기 때문에 256이상의 값을 표현하지 못하기 때문이다. 그래서 255, 256인 경우의 값이 더해지게 되어서 문제가 발생한 것을 알 수 있었다.

**텍스트, 편지이(가) 표시된 사진

자동 생성된 설명**

결과적으로 위와 같이 코드를 수정해서 문제를 해결할 수 있었다.

**6. 전체 소스코드**

|  |
| --- |
| function [feq, Heq] = heq(f, L)  % f: input image  % L: maximum intensity (for 8-bit gray-scale image, L = 255)  % feq: output image  % Heq: histogram of output image  % f\_H: histogram of input image  feq = f;  Heq = zeros(1,L + 1);  f\_H = zeros(L + 1, 1);  [height,width] = size(feq);  % histogram of input image without imhist function  for i = 1:height  for j = 1:width  %feq(i, j): uint8 -> Can't express 256  pixel\_value = cast(feq(i, j), 'double');  f\_H(pixel\_value + 1) = f\_H(pixel\_value + 1) + 1;  end  end  % Cumulative distribution of H  cH = cumsum(f\_H);  cH = round(cH \* L/cH(end)); % cH(end) = number of pixels  % output image  for i = 1:height  for j = 1:width  %feq(i, j): uint8 -> Can't express 256  pixel\_value = cast(feq(i, j), 'double');  feq(i,j) = cH(pixel\_value + 1);  end  end  % histogram of output image without imhist function  for i = 1:height  for j = 1:width  %feq(i, j): uint8 -> Can't express 256  pixel\_value = cast(feq(i, j), 'double');  Heq(pixel\_value + 1) = Heq(pixel\_value +1) + 1;  end  end  end |