IP Practice Report



[02] Histogram Equalization

2021/03/23

최현석

201702083

|  |
| --- |
| **INDEX**  1 - 구현 코드 ⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅ 3  2 - 결과 캡처 ⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅ 5  3 - 느낀 점 ⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅ 9    4 - 과제 난이도 ⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅⋅ 9  **1 - 구현 코드**   * **My\_calcHist(src)** * 주어진 src의 histogram을 구하는 함수이다. * 0 ~ 255의 인덱스를 갖는 histogram을 0으로 초기화하여 생성하였다. * Src의 모든 픽셀을 for문을 통해 돌며 0~255 Value 값을 구해 histogram의 해당 인덱스에 +1을 해주었다. * **My\_normalize\_hist(histogram, pixel\_num)** * My\_calcHist를 통해 구한 histogram 리스트의 모든 값을 pixel\_num으로 정규화하는 함수이다. * 반환할 Normalized를 생성하고 For 문을 통해 histogram의 모든 값을 pixel\_num으로 나눈 후 normalized에 넣어준다. * **My\_PDF2CDF(pdf)** * My\_normalize\_hist를 통해 정규화된 histogram을 누적하는 함수이다. * 반환할 cdf 를 생성하고 if 조건문을 통해 누적 값을 cdf에 넣어줬다. * cdf[0] = pdf[0] * cdf[n] = pdf[n-1] + pdf[n] * **My\_denormalize(normalized, gray\_level)** * 누적된 정규화 히스토그램을 다시 max\_gray\_level 곱해주는 함수이다. * 반환할 denormalized를 생성하고 for문을 통해 각 값에 gray\_level을 곱함 * **My\_calcHist\_equalization(denormalized, hist)** * 위 과정에서 구한 값들을 이용하여 histogram equalization을 하는 함수이다 * 반환할 hist\_equal을 0으로 초기화하고 for문을 통해 해당 값을 구해 누적한다. * Hist\_equal[ denormalized[i] ] += hist[ i ] * **My\_equal\_img(src, output\_gray\_level)** * 이미지에 histograml equalization을 적용하는 함수이다. * Mapping function을 보면 ( I, j ) 의 픽셀 value가 x 일 때 histogram\_equalization의 값은 output\_gray\_level[x] 이다. * dst[I,j] = output\_gray\_level[src[I,j]] * **My\_round(list)** * Astype(int) 에서 오류가 발생하여 직접 구현한 함수이다. * For문을 돌며 모든 값에 대해 int(,,,) 형 변환을 해서 버림하였다. * **Mapping function** * Hist\_equal = Integral 0 to x ( histogram ) 이므로 * X = range ( 0 , 256 ) -> 0,1,2,3,4,…..,255 를 갖음 * Y ( 크기 256 0으로 초기화 )   -> x에 대해for문을 돌며  Y[x] ( = Integral 0 to x ( histogram) ) = output\_gray\_level[x]  **2 – 결과 캡쳐**  **( Src = fruits\_div3.jpg )**  **My\_calcHist(src)**    **My\_hist\_equal(src, …)**    **Mapping function**    **Fruits\_div3.jpg and histogram**      **Equalization image and histogram**       1. **– 느낀 점**  * Hist\_equal 을 통해 value 가 몰려있어 흐릿했던 이미지가 평활화 과정을 거치면서 예상했던대로 value가 퍼져 대비가 높아져 선명해졌다. * 보통 C나 Java를 하면 데이터 -> 데이터 꼴로 입력과 출력이 이루어졌다. 그런데 이번 과제는 이미지를 pixel 단위로 쪼개고 value를 직접 수정하면서 결과를 눈으로 볼 수 있어서 흥미롭다. * 다양한 편집 기법에 사용되는 mapping function을 좀 더 찾아보고 적용해보고 싶단 생각이 들었다.   **4 – 과제 난이도**   * 이론 수업 때 각 값이 뜻하는 바와 어떻게 구하는지 잘 들었다면 어렵지 않았다고 생각합니다.   **END** |

‘’’’’’’’’’’’’