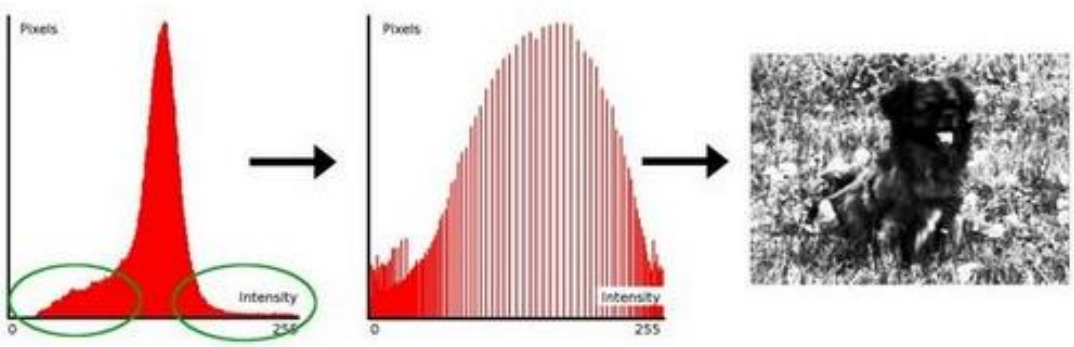
영상 처리 03차 과제

컴퓨터 공학과 17101244 조서연

1. Histogram Equalization



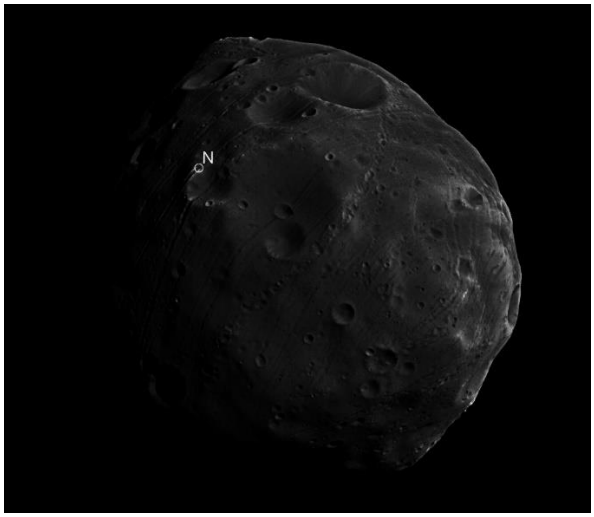
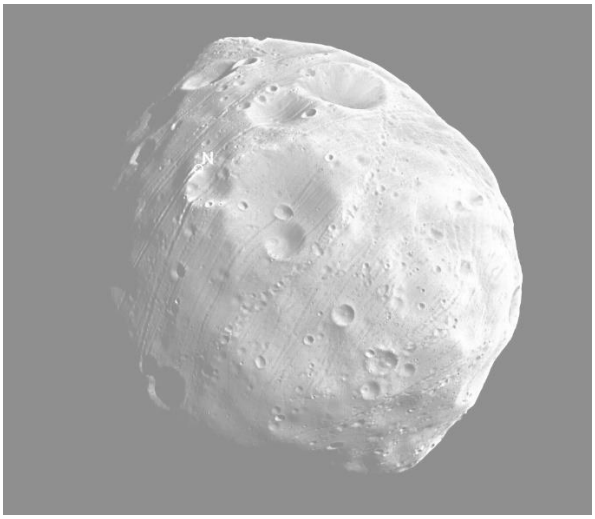
히스토그램 평준화 알고리즘은 흑백 이미지의 Intensity를 높여 이미지의 대비효과를 준다.

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j \quad k = 0,1,2,\ldots,L-1$$

다음과 같은 식을 가지고 평준화를 하며 식의 내용은 r강도를 가진 픽셀의 개수를 누적합한 것에 (L-1)/MN(size)라는 상수를 곱한 것이 Sk가 된다는 것이다 여기서 Sk는 명암 대비 보정을 시켜준 새로운 이미지 r강도를 가진 픽셀의 개수가 된다. 핵심 코드는 다음과 같다.

```
int size = image.rows * image.cols;
float alpha = 255.0 / size;

int sumhistogram[256];
sumhist(histogram, sumhistogram); // 시그마 nk

// Scale the histogram
int Sk[256];
for (int i = 0; i < 256; i++){
    Sk[i] = cvRound((double)sumhistogram[i] * alpha);
    //시그마(nk/size) * 255 시그마 반올림해준다.
}
```

알파는 (L-1)/MN(size) 상수를 뜻하며 sumhistogram은 누적 히스토그램 합이다.

위는 결과 이미지다.

2. Histogram Matching

입력 이미지가 타겟 이미지의 히스토그램을 갖도록 histogram matching을 수행한 결과 이미지

```
// 각 intensity 마다의 pixel 들을 count 해준다.
void imhist(Mat image, int histogram[]) {

    for (int i = 0; i < 256; i++) {
        histogram[i] = 0;
    }

    for (int y = 0; y < image.rows; y++)
        for (int x = 0; x < image.cols; x++)
            histogram[(int)image.at<uchar>(y, x)]++;
}
```

각 intensity 마다의 pixel들을 count를 해주는 imhist 함수

```
// sumHist 로 histgram 분포를 구한다 .
void sumhist(int histogram[], int sumhistogram[]) {
    sumhistogram[0] = histogram[0];
    for (int i = 1; i < 256; i++) {
        sumhistogram[i] = histogram[i] + sumhistogram[i - 1];
    }
}
```

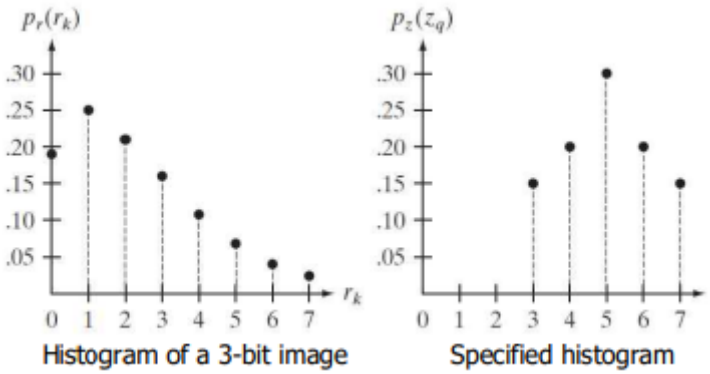
Sumhist로 histogram 분포를 구한다

```cpp
//cdf 를 구한다.
void CDF(Mat image, int cdf[]) {
    double sumHist[256];

    int histogram[256];
    imhist(image, histogram); // 이미지 히스토그램 분포 확인

    int size = image.rows * image.cols;
    float alpha = 255.0 / size;

    int sumhistogram[256];
    sumhist(histogram, sumhistogram); // 시그마 nk

    for (int i = 0; i < 256; i++) {
        cdf[i] = cvRound((double)sumhistogram[i] * alpha);
        //시그마(nk/size) * 255 시그마 반올림해준다.
    }
}
```

이미지의 히스토그램 분포를 확인하고, 누적 히스토그램에 alpha(255/넓이)를 곱해서 CDF를 구한다

CDF(image_input, PI); // 입력 이미지의 CDF 만들기

CDF(image_target, PJ); // 타겟 이미지의 CDF 만들기



Histogram of a 3-bit image　　Specified histogram

```
Mat Matching(Mat image_input, Mat image_target) {
    int PI[256], PJ[256];
    CDF(image_input, PI);
    CDF(image_target, PJ);
    Mat image_output = image_input.clone();

    int LUT[256];
    for (int i = 0; i < 256; ++i) LUT[i] = 0; //Lookup Table 초기화
    for (int i = 0; i < 256; ++i) {
        for (int j = 1; j < 256; ++j) {
            if (PI[i] > PJ[j - 1] && PI[i] <= PJ[j]) LUT[i] = j; // 가까운 걸로 넣는다.
        }
    }
    for (int i = 0; i < image_input.rows; ++i) {
        for (int j = 0; j < image_input.cols; ++j) {
            image_output.at<uchar>(i, j) = LUT[image_input.at<uchar>(i, j)]; // 이미지 처리
        }
    }
    return image_output;
}
```

해당 하는 부분은

int LUT[256];

for (int i = 0; i < 256; ++i) LUT[i] = 0; //Lookup Table 초기화

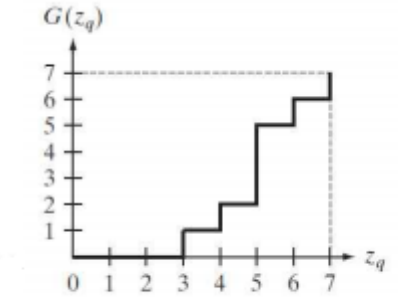
for (int i = 0; i < 256; ++i) {

for (int j = 1; j < 256; ++j) {

if (PI[i] > PJ[j - 1] && PI[i] <= PJ[j]) LUT[i] = j; // 가까운 걸로 넣는다.

}

}



Transformation function obtained from the specified histogram

s_k	→	z_q
1	→	3
3	→	4
5	→	5
6	→	6
7	→	7

Mappings of s_k into corresponding z_q

이미지의 Lookup table 을 구성한다

입력 이미지의 CDF와 타겟 이미지의 CDF를 구하고 출력 이미지를 구하기 위해 LUT를 구성한다. 0을 제외하고 P[j-1](타겟 이미지) < PI[i](입력 이미지) <= P[j](타겟 이미지) 로 사이 값에 있는 구간을 찾아 LUT 테이블을 구성한다. 각 강도에 대한 LUT 테이블을 구성한 다음, 이를 이용하여 해당하는 입력 이미지 위치에서의 출력 이미지를 얻어낸다.

왼쪽 그림은 원본 이미지, 오른쪽 그림은 결과 이미지이다. Target 이미지는 그 아래 이미지와 같고,

원본 이미지에서는 명암이 높아 보이지 않던 부분이 target 이미지를 가지고 image matching 후에는 윤곽이 뚜렷해진 것을 확인 할 수 있다.