Quantization

Uniform scalar quantizer

과목명	디지털신호처리
담당교수	심동규 교수님
학과	컴퓨터정보공학부
학년	3학년
학번	2019202009
이름	서여지
제출일	2021.10.24(일)

KWANGWOON UNIVERSITY

1. 과제 개요

이번 과제에서는 C++ 언어를 이용해 Uniform scalar 양자화기와 역양자화기를 구현하는 것을 목표로 한다. 주어진 테스트 이미지에 대해 Loss를 최소화하는 양자화기를 설계한다. 테스트 이미지는 512 * 512 해상도를 갖고, R, G, B 채널로 이루어져 있다. 각각의 채널에 대해 한 픽셀에는 8bit의 색상 정보가 들어있다. 8bit보다 적은 수의 bit를 이용하여 이미지를 저장하는 양자화기와, 저장한 이미지를 다시 복원하는 역양자화기를 설계한다.

2. 과제 수행 방법 (이론과 구현)

- Uniform scalar Quantization

  양자화는 전체 범위의 샘플 값을 유한한 수의 심볼로 매핑하는 과정이다. 신호 값의 전체 범위를 유한한 수의 구간으로 구분하고, 동일한 구간의 샘플에는 동일한 심볼로 매핑한다. 양자화 오차는 이 과정에서 발생한 양자화 결과와 원본의 차이를 의미한다.

  Uniform scalar Quantization은 다수의 샘플을 모두 독립적으로, 동시에 균일한 간격으로 나누어 양자화 한 것을 의미한다. Uniform scalar Quantization은 가장 단순한 양자화 방법이며 샘플 값이 균일하게 분포할 때 최적의 결과를 얻는다. 샘플이 갖는 전체 값의 범위를 구간의 수로 나누어 $\frac{\textit{전체 값의 범위}}{\textit{양자화 구간의 수}}$ 크기의 구간을 이용한다. 각 구간에 매핑되는 값은 구간의 중간 값을 이용하였다. 이는 $\textit{구간의 첫 번째 값} + \frac{\textit{구간의 크기}}{2}$로 얻을 수 있다.

- 구현방법

  - raw구조

    제시된 raw파일은 한 픽셀이 R, G, B 총 3개의 채널에 대한 값을 갖는다. raw 파일에 세 개의 채널이 저장된 순서를 두 가지 방법으로 추측할 수 있었다. 한 픽셀에 대한 R, G, B 채널의 값이 번갈아가며 쓰인 구조 혹은 R 채널에 대한 정보가 먼저 쓰여있고, 다른 채널의 정보가 이어서 쓰인 구조로 추측하였다. 이것을 확인하기 위해 원본 이미지를 xSize*(ySize*3) 만큼 읽은 뒤, xSize * ySize만큼 파일로 출력하여 확인하였다. 그 결과 R 채널에 대한 정보가 저장된 것을 확인할 수 있었고, 두 가지 경우 중 두 번째 경우에 해당함을 알 수 있었다.

raw파일의 구조와 확인결과 출력된 R채널의 이미지

■ 제시된 그림 특징

8bit 파일은 0에서 255사이의 값을 갖지만, 제시된 이미지가 실제로 사용하는 값의 최소값과 최댓값을 확인하였다. 제시된 이미지는 3에서 255의 값을 가졌다. 0에서 2까지 3개의 값을 제외하고 양자화 하는 것이 약간의 오차를 줄일 수 있을 것으로 예상할 수 있다. 그러나 처리 과정에 비해 줄어드는 오차가 미미할 것으로 예상하여 0에서 255 범위의 값을 균일한 구간으로 나누어 양자화 하였다.

```
//양자화
//기존 8bit - 0~255

//이미지에서 사용하는 최소, 최댓값 구간 알아보기

int min = 255, max = 0;

for (int i = 0; i < xSize; i++) {
    for (int j = 0; j < ySize; j++) {
        min = (image[i][j] < min) ? image[i][j] : min;
        max = (image[i][j] > max) ? image[i][j] : max;
    }
}

cout << "min: " << min<<"₩tmax: " << max << endl;
```

```
Microsoft Visual Studio 디버그 콘솔
min: 3  max: 255
```

■ 프로그램 구조

◆ 양자화기

양자화기는 input 디렉토리의 Lenna_512x512_original.raw를 읽어서 양자화 한 뒤, output 디렉토리를 생성하여 Lenna_512x512_code로 출력한다.

프로그램은 원본 파일을 읽는 부분, 양자화 하는 부분, 결과를 저장하는 부분으로 나누어진다. 동적할당한 unsigned char 배열을 이용하여 원본 파일을 모두 읽은 뒤, 배열의 값을 구간의 크기로 나누는 방법으로 양자화 한다. 양자화 구간의 크기는 사용하는 bit수 N에 따라 변화하며, $\frac{256}{2^N}$ 값을 갖는다. 결과를 code파일에 저장하는 과정은 unsigned int 형의 버퍼를 이용하여 수행된다. 양자화된 값을 버퍼에 N bit씩 채워넣고, 버퍼의 상위 1byte를 fputc함수를 이용하여 파일에 쓰는 과정을 반복한다.

◆ 역양자화기

역양자화기는 input 디렉토리의 Lenna_512x512_code를 읽어서 역양자화 한 뒤, output 디렉토리를 생성하여 Lenna_512x512_reconstruct.raw로 출력한다.

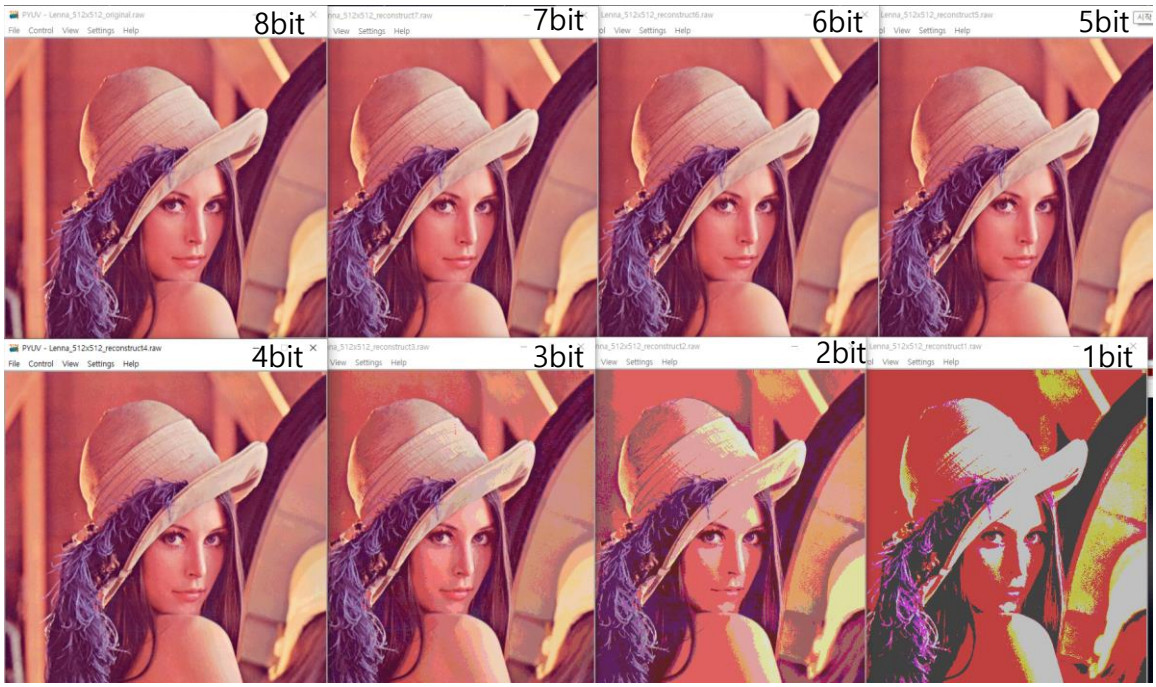
프로그램은 code파일을 읽는 부분, 역양자화 하는 부분, 결과를 저장하는 부분으로 나누어진다. fgetc함수를 이용하여 code 파일의 값을 1byte씩 읽어 unsigned int형의 버퍼에 저장한다. 버퍼의 상위 N 비트를 동적할당한 unsigned char 배열에 저장하여 양자화 코드를 읽는다. 이후 양자화기에서와 동일한 방법으로 구간의 크기를 구한 뒤, 배열의 값을 역양자화 한다. 이때 $\textit{역양자화 결과} = \textit{양자화 코드} * \textit{구간의 크기} + \left(\frac{1}{2} * \textit{구간의 크기}\right)$ 임을 이용한다. 양자화코드*구간의크기 결과는 양자화 구간의 가장 작은 수를 의미하므로 구간의 크기의 절반에 해당하는 값을 더해서 구간의 중간 값을 구할 수 있다. 이후 역양자화 결과를 파일에 쓴 뒤 종료한다.

◆ Loss 계산기

Loss 계산기는 input 디렉토리의 Lenna_512x512_original.raw, Lenna_512x512_reconstruct.raw, Lenna_512x512_code를 읽어서 code size, bit per pixel, MSE, Loss 계산 결과를 화면에 출력한다.

code file을 읽어서 codeSize를 얻고, codeSize를 전체 픽셀의 수로 나누어 bitPerPixel값을 계산한다. 원본 이미지와 역양자화 결과 이미지의 차를 제곱한 값을 (전체 픽셀 수 * 채널 수)로 나누어 MSE를 구한다. 마지막으로 Loss는 bitPerPixel값에 MSE*4 값을 더하여 구하였다.

3. 결과



bit	구간크기	codesize(bit)	bit per pixel	MSE	Loss
8	1	6291456	24	0	24
7	2	5505024	21	0.492086	22.9683
6	4	4718592	18	1.50674	24.027
5	8	3932160	15	5.54247	37.1699
4	16	3145728	12	22.0863	100.345
3	32	2359296	9	88.8075	364.23
2	64	1572864	6	330.474	1327.9
1	128	786432	3	1219.01	4879.06

제시된 이미지에 대해 1~7bit를 이용해 양자화 한 뒤, 다시 복원한 결과는 위와 같이 나타났다. Loss 평가가 가장 뛰어난 것은 7bit를 이용하여 양자화 한 경우였으며, 이 경우 원본 영상보다도 효과적인 것으로 나타났다. 6bit를 이용하면 원본인 8bit를 이용한 경우와 비슷한 Loss를 보였다. 그 외의 경우엔 bit수가 줄어들며 MSE의 값이 큰 차이로 증가하여 양자화 오차가 크게 나타나는 것을 확인할 수 있었다.

4. 고찰

8bit를 이용한 원본보다 7bit를 이용한 결과의 Loss 평가 값이 더 작은 부분이 흥미로웠다. 7bit로 원본 이미지를 양자화 한다면 하나의 구간은 두 개의 값을 갖고, 이것은 두 가지 수를 하나의 수로 매핑한 것과 같으므로 양자화 오차는 0또는 1로 나타날 것이다. 만약 모든 값에 대해 양자화 오차가 1로 나타나는 극단적인 경우 MSE가 1이 되어 Loss를 계산하였을 때 21+ 1*4 = 25 값을 갖을 수 있지만, 구간 내의 값의 분포가 충분히 균등한 경우 원본의 Loss 값인 24보다 낮은 값을 가질 수 있다는 것을 확인할 수 있다. 이 경우 7bit를 이용한 양자화의 MSE가 0.49 정도로 나타난 것을 통해 제시된 이미지는 두 개의 값을 갖는 구간에서 충분히 균등하게 나타난 것을 확인할 수 있다.

또한 다른 bit수를 이용하여 양자화 한 경우에도 각 구간에서 대체로 균등한 값을 갖는 것을 알 수 있다.

bit	구간 크기	MSE		
8	1	0		
7	2	0.492086	$0^2 * \frac{1}{2} + 1^2 * \frac{1}{2}$	0.5
6	4	1.50674	$((-1)^2 + 0^2 + 1^2 + 2^2) * \frac{1}{4}$	1.5
5	8	5.54247	$((-3)^2 + (-2)^2 + (-1)^2 + 0^2 + 1^2 + 2^2 + 3^2 + 4^2) * \frac{1}{8}$	5.5
4	16	22.0863	$(2 * (1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 + 7^2) + 8^2 + 0^2) * \frac{1}{16}$	21.5