Quantization

Non-uniform vector quantizer

과목명	디지털신호처리
담당교수	심동규 교수님
학과	컴퓨터정보공학부
학년	3학년
학번	2019202009
이름	서여지
제출일	2021.11.9 (화)

KWANGWOON UNIVERSITY

1. 과제 개요

저번 과제에 이어서 C++ 언어를 이용하여 양자화기를 구현한다. 이번에 구현하는 양자화기와 역양자화기는 non-uniform vector 방식의 양자화와 역양자화를 수행한다. 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)

- Non-uniform vector Quantization

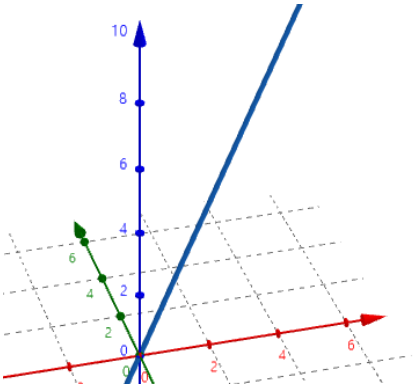
  non-uniform vector Quantization은 다수의 샘플을 벡터로 지정하여, 균일하지 않은 간격으로 나누어 양자화 한 것을 의미한다. training data의 값에 따라 양자화하기 위한 구간의 크기와 구간에 매핑되는 벡터가 변화한다. 이번 과제에서는 한 픽셀 지점에서의 R, G, B 세 채널의 값을 사용한 3차원 벡터를 이용하여 양자화를 수행하였다.

  초기 reconstructed vector를 지정하고, 나머지 vector를 reconstructed vector와의 거리에 따라 가까운 vector의 그룹으로 분류한다. 그룹으로 분류된 벡터들의 평균을 계산하여 새로운 reconstructed vector를 지정한다. vector를 그룹으로 분류하는 과정과 그룹의 대푯값을 계산하는 과정을 여러 번 반복하여 양자화한다.

- 구현방법

  - 초기 vector

    무작위로 2^N개의 vector를 지정하여 초기 그룹을 분류한다. 이번 과제에서는 vector에 전체 구간을 2^N개로 나눈 값을 이용해서 x,y,z을 모두 같은 값으로 지정하여 사용하였다.



(초기 벡터 형태)

■ 프로그램 구조

◆ 양자화기

양자화기는 input 디렉토리의 Lenna_512x512_original.raw를 읽어서 양자화 한 뒤, output 디렉토리를 생성하여 Lenna_512x512_code와 Lenna_512x512_codebook을 출력한다.

프로그램은 원본 파일을 읽는 부분, 초기 벡터의 값을 구하는 부분, 벡터를 분류하는 것과 새로운 reconstructed vector를 구하는 것을 반복하며 계산하여 양자화하는 부분, 결과를 저장하는 부분으로 나누어진다. 3차원 벡터를 저장하기 위한 $2^N * 3$크기의 v 배열로 동적할당하여 사용한다. 초기 벡터값을 지정한 이후 반복문 내부에서 image의 각 픽셀에 해당하는 벡터를 v배열의 벡터 중 가장 가까운 벡터 그룹으로 분류한다. 분류된 결과는 code 배열에 저장된다. 분류한 그룹내의 벡터들의 평균값으로 v배열이 수정된다. 이 과정은 v배열의 x값이 변경된 벡터가 하나도 존재하지 않거나, 20번 반복했을 때 종료된다. 이후 code 배열의 값을 code파일에 출력하고, v배열의 값을 codefile로 출력한다. 결과를 code파일에 저장하는 과정은 저번 과제 구현에서와 동일하게 unsigned int형의 버퍼를 이용하여 수행된다. 하위 Nbit를 채워넣고, 상위 1byte를 쓰는 과정을 반복한다.

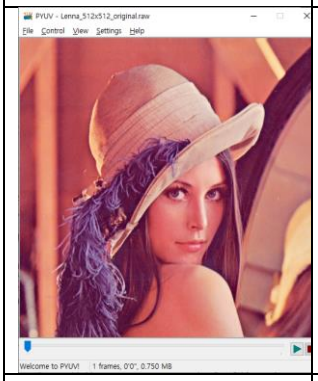
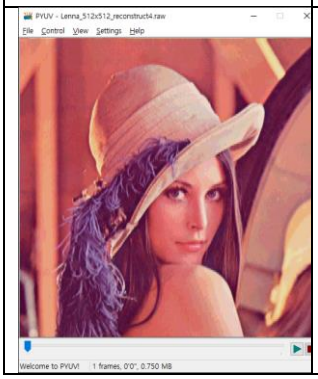
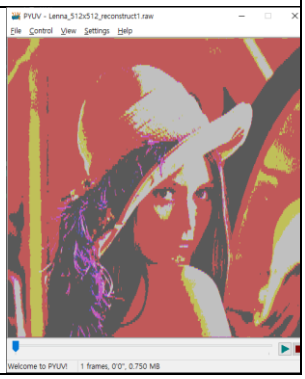
◆ 역양자화기

역양자화기는 input 디렉토리의 Lenna_512x512_code과 Lenna_512x512_codebook을 읽어서 역양자화 한 뒤, output 디렉토리를 생성하여 Lenna_512x512_reconstruct.raw로 출력한다.

프로그램은 code와 codebook 파일을 읽는 부분, 역양자화 하는 부분, 결과를

저장하는 부분으로 나누어진다. code파일을 읽는 부분은 저번 과제와 동일하게 파일의 값을 1byte씩 읽어 버퍼에 저장한 뒤, 상위 N 비트를 image 배열에 저장하여 진행된다. codebook파일의 내용은 양자화 과정에서 사용한 것과 동일한 구조의 배열 v에 저장한다. image[i][j]의 값(k)과 대응하는 v배열의 벡터(v[k][0], v[k][1], v[k][2])를 이용하여 역양자화한 뒤 결과를 파일에 쓰고 종료한다.

3. 결과

<table>
<tr><td>8bit(original)</td><td>7bit</td><td>6bit</td><td>5bit</td></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
<tr><td>4bit</td><td>3bit</td><td>2bit</td><td>1bit</td></tr>
<tr><td>  </td><td>  </td><td>  </td><td>  </td></tr>
</table>

bit	codesize(bit)	bit per pixel	MSE	Loss
8	6291456	24	0	24
7	1835008	7	16.5136	73.0544
6	1572864	6	24.5831	104.332
5	1310720	5	37.8929	156.572
4	1048576	4	65.6547	226.619
3	786432	3	124.348	500.392
2	524288	2	330.725	1324.9
1	262144	1	958.879	3836.52

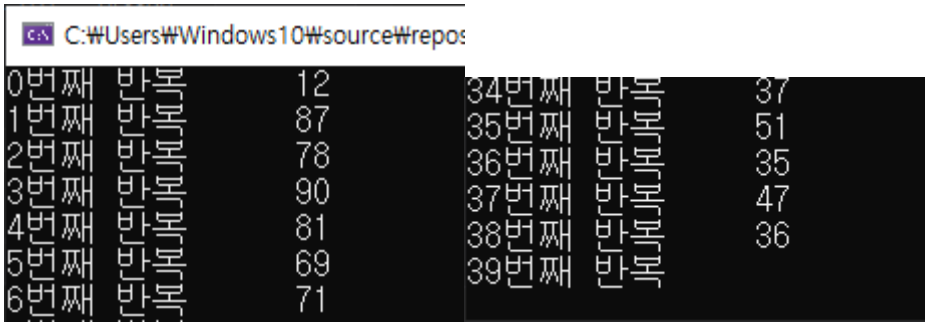
제시된 이미지에 대해 1~7bit를 이용해 양자화 한 뒤, 다시 역양자화한 결과는 위와 같이 나타났다. Loss 평가가 가장 뛰어난 것은 이전 과제에서와 마찬가지로 7bit를 이용하여 양자화 한 경우였다. 이번에 작성한 vector quantizer는 R, G, B 세 채널의 정보를 하나의 값으로 저장하기 때문에 bit per pixel의 크기가 정보를 저장할 때 사용한 bit수와 동일하게 나타났다. 또한 한 단계 적은 bit를 이용했을 때 MSE의 증가량이 약 1.49배에서 2.89배까지 증가하는 것을 확인할 수 있다.

4. 고찰

저번 과제에서 구현한 scalar quantizer와 결과를 비교하면 다음과 같다.

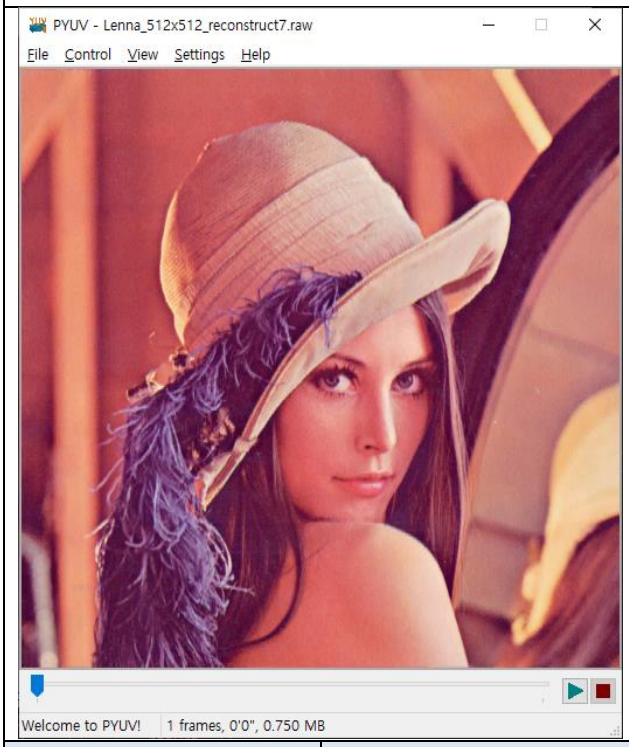
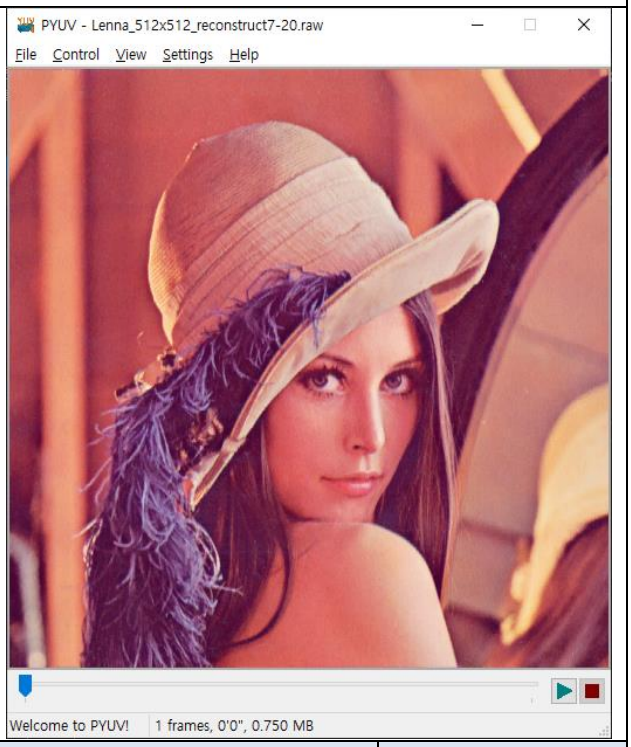
	Non-uniform vector quantizer		scalar uniform quantizer		scalar non-uniform quantizer	
bit	MSE	Loss	MSE	Loss	MSE	Loss
8	0	24	0	24	0	24
7	16.5136	73.0544	0.492086	22.9683	0.483655	22.9346
6	24.5831	104.332	1.50674	24.027	1.42706	23.7083
5	37.8929	156.572	5.54247	37.1699	5.25129	36.0052
4	65.6547	226.619	22.0863	100.345	19.0716	88.2865
3	124.348	500.392	88.8075	364.23	65.3006	270.201
2	330.725	1324.9	330.474	1327.9	227.406	1115.62
1	958.879	3836.52	1219.01	4879.06	926.297	3708.19

vector quantizer는 scalar quantizer와 bit per pixel의 차이가 있기 때문에 3bit, 6bit vector quantization결과와 1bit, 2bit scalar quantization결과를 비교한다. 위 결과에서 vector quantizer가 scalar quantizer보다 뛰어난 성능을 보인 것을 확인할 수 있다. MSE를 비교했을 때 bit per pixel이 4인 vector quantizer와 9인 3bit scalar non-uniform quantizer의 결과가 유사했다. 따라서 vector quantizer는 이 경우에 scalar quantizer보다 적은 bit를 이용하여 더 높은 품질을 보존하는 양자화가 가능했다.



벡터의 구간을 분류하고, 분류한 것을 바탕으로 다시 벡터를 계산하는 반복의 수에 따라 결과가 다르게 나타난다. 적은 bit를 사용했을 때는 더 이상 벡터의 변경이 없을 때까지 반복하는 것이 가능했지만, 많은 bit를 사용한 양자화에서는 반복하는 수가 증가하며 계산에 많은 시간이 소요되었다.

7bit를 이용한 vector quantizer에서 양자화 과정을 50번, 20번 수행한 결과와 MSE, Loss는 다음과 같다.

7bit (50회 반복)	7bit (20회 반복)
	

bit per pixel	iteration	MSE	Loss
7	50	16.5136	73.0544
7	20	17.0737	75.295

이미지 출력 결과 두 경우의 차이가 크게 확인되지 않았다. 그러나 MSE계산 결과 벡터를 분류하고 새로운 벡터를 계산하는 과정을 많이 반복했을 때의 MSE값이 더 적고, 따라서 Loss또한 감소하는 것을 확인할 수 있다. 계산에 사용되는 시간과 그 결과 얻을 수 있는 정확도의 이득을 응용에 따라 조정할 필요가 있을 것이다.