Yingjie Lian
CS 4600
September 15, 2018

HW1 - DCT

Audio Coding

For this part, I followed the equations and directions in the pdf. Also, I have looked through the given lecture and discussion board. Here are the screenshots for all three methods I wrote.

DCT

```
void DCT(const float* A, float* C, int size)
{
    // TODO: part of Homework Task 1
    // takes a vector A and produce as output a vector C, the 1D Discrete Cosine Transform
    // Use std::cos

    for (int u = 0; u < size; u++) {
        double sum = 0.;
        double c = (u == 0) ? (sqrt(2)/4) : (1/2);

        for (int i = 0; i < size; i++) {
            sum += std::cos((2*i+1)*u*M_PI / (2*size)) * A[i];
        }
        C[u] = c * sum;
    }
}
```

For the DCT method, I followed closely to the equation 1. Since it was a 1D DCT, not very hard to complete by filling the parameters and inputs to the right positions of the equation.

Compress

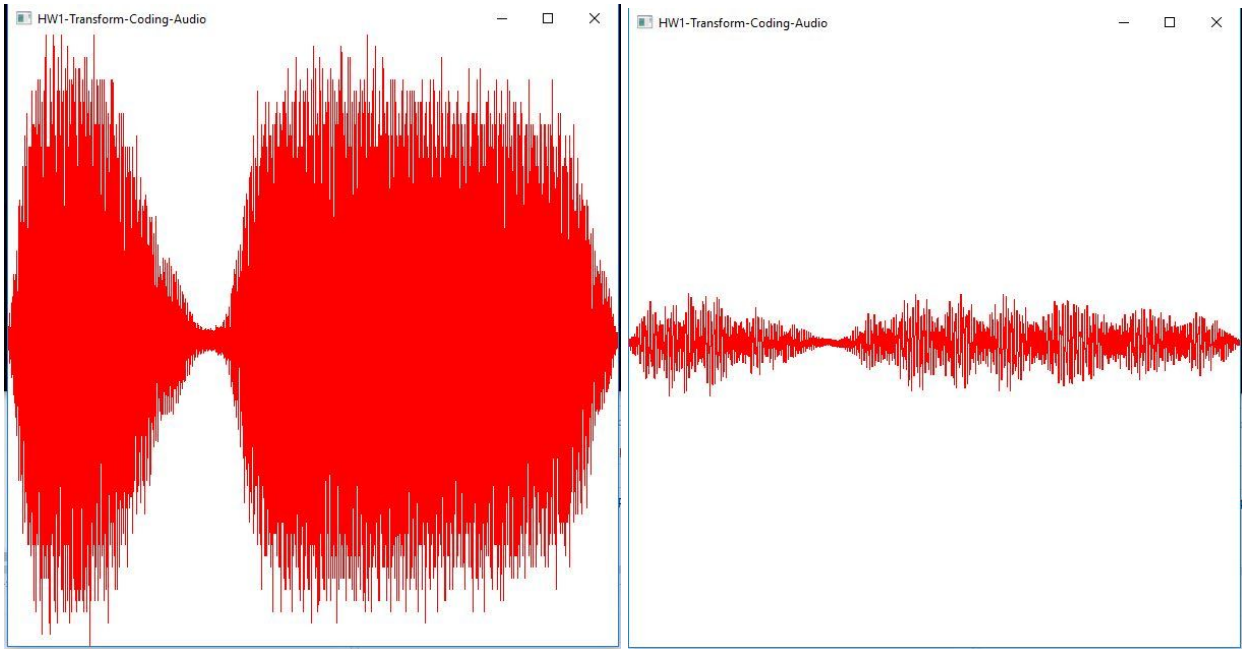
```
void compress(float* C, int size, int m)
{
    // TODO: part of Homework Task 1
    // sets last m element as zero

    for (int i = (size - m); i < size; i++)
    {
        C[i] = 0;
    }
}
```

This one is a little bit tricky because it said zeros out last m elements of C. So when I was coding for that, I have to count the number from (size - m), not from m to the end.

inverseDCT

```
void inverseDCT(const float* C, float* B, int size)
{
    // TODO: part of Homework Task 1
    // takes a vector C and produce as output a vector B, the 1D inverse Discrete Cosine Transform
    // Use std::cos

    for (int i = 0; i < size; i++)
    {
        double sum = 0;

        for (int u = 0; u < size; u++)
        {
            double c = (u == 0) ? (sqrt(2) / 4) : (1 / 2);
            sum += c * std::cos((2 * i + 1)*u*M_PI / (2 * size)) * C[u];
        }
        B[i] = sum;
    }
}
```

The inverse part is also not hard to complete. Once you finished the DCT part, inverse will be always easy. But, one thing need to pay attention to is the positions of inputs such as i and u in this method.



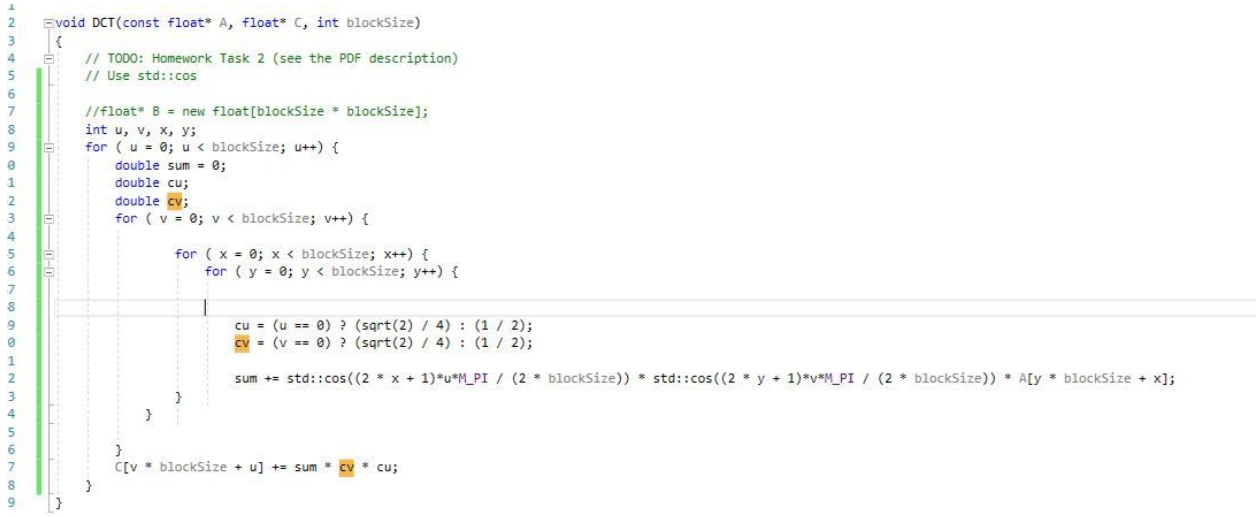
(Original) (Compression)

The tone of the original one sounds very sonorous, but the compressed audio sounds kind of deep and low. Actually, you can tell by seeing the images of those two audio sound waves.

Image Coding

Had some hard time for thinking how to deal with the 2D DCT unless I saw processImage method in the class. Pretty much the same for 2D DCT, just be careful to deal with the sum part. In addition, the discussion board helped me a lot.

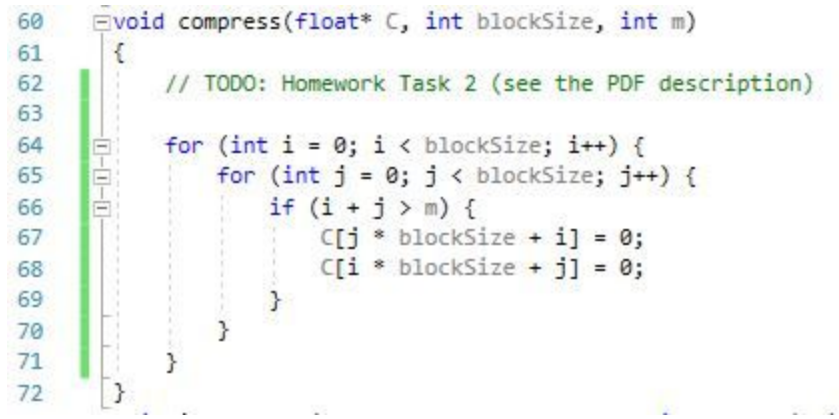
DCT



```
void DCT(const float* A, float* C, int blockSize)
{
    // TODO: Homework Task 2 (see the PDF description)
    // Use std::cos

    //float* B = new float[blockSize * blockSize];
    int u, v, x, y;
    for ( u = 0; u < blockSize; u++) {
        double sum = 0;
        double cu;
        double cv;
        for ( v = 0; v < blockSize; v++) {

                for ( x = 0; x < blockSize; x++) {
                    for ( y = 0; y < blockSize; y++) {

                        cu = (u == 0) ? (sqrt(2) / 4) : (1 / 2);
                        cv = (v == 0) ? (sqrt(2) / 4) : (1 / 2);

                        sum += std::cos((2 * x + 1)*u*M_PI / (2 * blockSize)) * std::cos((2 * y + 1)*v*M_PI / (2 * blockSize)) * A[y * blockSize + x];
                }
            }

        }
        C[v * blockSize + u] += sum * cv * cu;
    }
}
```

This function is the heart of what I had to do for the compression and uncompression. The most difficult part for me is thinking how to use 1D array deal with 2D DCT since A and C cannot represent 2D array. The rest of others are pretty the same with 1D DCT in audio coding.

Compress



```
void compress(float* C, int blockSize, int m)
{
    // TODO: Homework Task 2 (see the PDF description)

    for (int i = 0; i < blockSize; i++) {
        for (int j = 0; j < blockSize; j++) {
            if (i + j > m) {
                C[j * blockSize + i] = 0;
                C[i * blockSize + j] = 0;
            }
        }
    }
}
```

This one is a little different with audio coding part, It is i+j>m instead of last m items. But the logic is the same. Use 2 for loops and 1 if statement to make sure zero out needed items.

inverseDCT

```
void inverseDCT(const float* C, float* B, int blockSize)
{
    // TODO: Homework Task 2 (see the PDF description)
    // Use std::cos
    int u, v, x, y;
    for (x = 0; x < blockSize; x++) {
        double sum = 0;
        double cu;
        double cv;
        for (y = 0; y < blockSize; y++) {

            for (u = 0; u < blockSize; u++) {
                for (v = 0; v < blockSize; v++) {

                    cu = (u == 0) ? (sqrt(2) / 4) : (1 / 2);
                    cv = (v == 0) ? (sqrt(2) / 4) : (1 / 2);

                    sum += cu* cv * std::cos((2 * x + 1)*u*M_PI / (2 * blockSize)) * std::cos((2 * y + 1)*v*M_PI / (2 * blockSize)) * C[u * blockSize + v];
                }
            }

        }
        B[x * blockSize + y] += sum;
    }
}
```

The inverse part is also not hard to complete. Once you finished the DCT part, inverse will be always easy. But, one thing need to pay attention to is the positions of inputs such as i and u in this method.

(Original)

(Decompressed)

As you can see, the decompressed version of the image is more fuzzy than the original image.