

Van Euclid Dy

CS 4600

DCT

```
float cu, cv, sum;
for (int u = 0; u < blockSize; u++) { //DCT calculation for 2-D
    for (int v = 0; v < blockSize; v++) {
        if (u == 0) { //determines c(u)
            cu = sqrtf(8) / 8;
        }
        else {
            cu = sqrtf(2) / sqrtf(8);
        }
        if (v == 0) { //determines c(v)
            cv = sqrtf(8) / 8;
        }
        else {
            cv = sqrtf(2) / sqrtf(8);
        }
        sum = 0;
        for (int x = 0; x < blockSize; x++) { //Inner loop that calculates summation
            for (int y = 0; y < blockSize; y++) {
                sum += std::cos(((2 * x + 1) * u * M_PI) / (2 * blockSize)) * std::cos(((2 * y + 1) * v * M_PI) / (2 * blockSize)) * A[x * 8 + y];
            }
        }
        C[u * 8 + v] = (cu * cv) * sum; //Saving to C
    }
}
```

Screen shot above is my DCT body for 2-D since 1-D is just a simplified version of this code. The code is essentially what the formula given is in code.

```
float cu, cv, sum;
for (int x = 0; x < blockSize; x++) { //IDCT calculation for 2-D
    for (int y = 0; y < blockSize; y++) {
        sum = 0;
        for (int u = 0; u < blockSize; u++) { //Inner loop
            for (int v = 0; v < blockSize; v++) {
                if (u == 0) { //determines c(u)
                    cu = sqrtf(8) / 8;
                }
                else {
                    cu = sqrtf(2) / sqrtf(8);
                }
                if (v == 0) { //determines c(v)
                    cv = sqrtf(8) / 8;
                }
                else {
                    cv = sqrtf(2) / sqrtf(8);
                }
                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 * 8 + v]);
            }
        }
        B[x * 8 + y] = sum; //Saves to B
    }
}
```

Screen shot above is my IDCT body for 2-D since 1-D is just a simplified version of this code. The code is essentially what the formula given is in code.

```


m = 3
FROM C Compress
C[0 0] = 4.773528
C[0 1] = 0.028254
C[0 2] = 0.012654
C[0 3] = -0.007326
C[0 4] = 0.000000
C[0 5] = 0.000000
C[0 6] = 0.000000
C[0 7] = 0.000000
C[1 0] = -0.062670
C[1 1] = -0.006547
C[1 2] = 0.010285
C[1 3] = 0.000000
C[1 4] = 0.000000
C[1 5] = 0.000000
C[1 6] = 0.000000
C[1 7] = 0.000000
C[2 0] = -0.019161
C[2 1] = 0.002789
C[2 2] = 0.000000
C[2 3] = 0.000000
C[2 4] = 0.000000
C[2 5] = 0.000000
C[2 6] = 0.000000
C[2 7] = 0.000000
C[3 0] = 0.016179
C[3 1] = 0.000000

```



Here is an example of compression at $m = 11$. I've set my spectrum to go as follows, blocksize + blocksize - 2 - m . When $m = 0$ the IDCT returns the original signal.

```
m = 0
FROM C Compress
C[0 0] = 4.773528
C[0 1] = 0.000000
C[0 2] = 0.000000
C[0 3] = 0.000000
C[0 4] = 0.000000
C[0 5] = 0.000000
C[0 6] = 0.000000
C[0 7] = 0.000000
C[1 0] = 0.000000
C[1 1] = 0.000000
C[1 2] = 0.000000
C[1 3] = 0.000000
```



Here is an example at max compression where $m = 14$. Notice how the only C is just at 0,0 anything else would just be black.