영상처리 HW #4

Due on 10 June

The purpose of this assignment is to implement DCT algorithm for a given still image. In order to evaluate the performance, we can compare the subjective quality reconstructed by inverse transform as well as the objective criterion such as MSE (mean square error). Furthermore, we can exploit the frequency spectrum in which the energy compaction performance could be compared. For this purpose, carry out the following assignments.

1. For the given 512x512 image 'Lena',
   (a) Perform 8x8 forward DCT and plot the frequency spectrum on the monitor in proper scale for easy observation.
   (b) Perform 8x8 inverse DCT to obtain the reconstructed image. Compute the MSE.

2. For the given 512x512 image 'Boat',
   (a) Perform 8x8 forward DCT and plot the frequency spectrum on the monitor in proper scale for easy observation.
   (b) Perform 8x8 inverse DCT to obtain the reconstructed image. Compute the MSE.

3. Discuss the results.

You can refer to the following 8x8 forward and inverse DCT algorithm.

(You can also build your own source code to perform the same DCT task.)

```
#define B_size 8
#define   nint(x)        ( (x)<0. ? (int)((x)-0.5) : (int)((x)+ 0.5) )
/*------------
 8x8 block DCT
 ------------*/
dct8x8 (ix)
int ix[][B_size];

{
        static float pi=3.141592653589793238;
        float x[B_size][B_size],z[B_size][B_size],y[B_size],c[40],s[40],
              ft[4],fxy[4],yy[B_size],zz;
        int i,ii,jj;

        for (i=0; i<40; i++) {
            zz = pi * (float)(i+ 1) / 64.0;
            c[i] = cos(zz);
            s[i] = sin(zz);
        }

        for (ii=0; ii<B_size; ii++)
            for (jj=0; jj<B_size; jj++)
                x[ii][jj] = (float)ix[ii][jj];

        for (ii=0; ii<B_size; ii++) {
            for (jj=0; jj<B_size; jj++)
                y[jj] = x[ii][jj];

            for (jj=0; jj<4; jj++)
                ft[jj] = y[jj] +  y[7-jj];

            fxy[0] = ft[0] +  ft[3];
```

```
        fxy[1] = ft[1] +  ft[2];
        fxy[2] = ft[1] - ft[2];
        fxy[3] = ft[0] - ft[3];

        ft[0] = c[15] * (fxy[0] +  fxy[1]);
        ft[2] = c[15] * (fxy[0] - fxy[1]);
        ft[1] = s[7] * fxy[2] +  c[7] * fxy[3];
        ft[3] = -s[23] * fxy[2] +  c[23] * fxy[3];

        for (jj=4; jj<8; jj++)
                yy[jj] = y[7-jj] - y[jj];

        y[4] = yy[4];
        y[7] = yy[7];
        y[5] = c[15] * (-yy[5] +  yy[6]);
        y[6] = c[15] * (yy[5] +  yy[6]);

        yy[4] = y[4] +  y[5];
        yy[5] = y[4] - y[5];
        yy[6] = -y[6] +  y[7];
        yy[7] = y[6] +  y[7];

        y[0] = ft[0];
        y[4] = ft[2];
        y[2] = ft[1];
        y[6] = ft[3];
        y[1] = s[3] * yy[4] +  c[3] * yy[7];
        y[5] = s[19] * yy[5] +  c[19] * yy[6];
        y[3] = -s[11] * yy[5] +  c[11] * yy[6];
        y[7] = -s[27] * yy[4] +  c[27] * yy[7];

        for (jj=0; jj<B_size; jj++)
                z[ii][jj] = y[jj];

}
```

```
for (ii=0; ii<B_size; ii++) {
        for (jj=0; jj<B_size; jj++)
                y[jj] = z[jj][ii];

        for (jj=0; jj<4; jj++)
                ft[jj] = y[jj] + y[7-jj];

        fxy[0] = ft[0] + ft[3];
        fxy[1] = ft[1] + ft[2];
        fxy[2] = ft[1] - ft[2];
        fxy[3] = ft[0] - ft[3];

        ft[0] = c[15] * (fxy[0] + fxy[1]);
        ft[2] = c[15] * (fxy[0] - fxy[1]);
        ft[1] = s[7] * fxy[2] + c[7] * fxy[3];
        ft[3] = -s[23] * fxy[2] + c[23] * fxy[3];

        for (jj=4; jj<8; jj++)
                yy[jj] = y[7-jj] - y[jj];

        y[4] = yy[4];
        y[7] = yy[7];
        y[5] = c[15] * (-yy[5] + yy[6]);
        y[6] = c[15] * (yy[5] + yy[6]);

        yy[4] = y[4] + y[5];
        yy[5] = y[4] - y[5];
        yy[6] = -y[6] + y[7];
        yy[7] = y[6] + y[7];

        y[0] = ft[0];
        y[4] = ft[2];
        y[2] = ft[1];
        y[6] = ft[3];
        y[1] = s[3] * yy[4] + c[3] * yy[7];
        y[5] = s[19] * yy[5] + c[19] * yy[6];
```

```
                y[3] = -s[11] * yy[5] +  c[11] * yy[6];
                y[7] = -s[27] * yy[4] +  c[27] * yy[7];

                for (jj=0; jj<B_size; jj++)
                        y[jj] = y[jj] / 4.0;

                for (jj=0; jj<B_size; jj++)
                        z[jj][ii] = y[jj];
        }

        for (ii=0; ii<B_size; ii++)
                for (jj=0; jj<B_size; jj++)
                        ix[ii][jj] = nint(z[ii][jj]);
}

/*---------------------
 8x8 block inverse DCT
 ---------------------*/
idct8x8 (ix)

int ix[][B_size];

{
        static float pi=3.141592653589793238;
        float x[B_size][B_size],z[B_size][B_size],y[B_size],c[40],s[40],
                ait[4],aixy[4],yy[B_size],zz;
        int i,ii,jj;

        for (i=0; i<40; i++) {
                zz = pi * (float)(i+ 1) / 64.0;
                c[i] = cos(zz);
                s[i] = sin(zz);
        }

        for (ii=0; ii<B_size; ii++)
                for (jj=0; jj<B_size; jj++)
```

```
            x[ii][jj] = (float)ix[ii][jj];

for (ii=0; ii<B_size; ii++) {
        for (jj=0; jj<B_size; jj++)
            y[jj] = x[jj][ii];

        ait[0] = y[0];
        ait[1] = y[2];
        ait[2] = y[4];
        ait[3] = y[6];

        aixy[0] = c[15] * (ait[0] + ait[2]);
        aixy[1] = c[15] * (ait[0] - ait[2]);
        aixy[2] = s[7] * ait[1] - s[23] * ait[3];
        aixy[3] = c[7] * ait[1] + c[23] * ait[3];

        ait[0] = aixy[0] + aixy[3];
        ait[1] = aixy[1] + aixy[2];
        ait[2] = aixy[1] - aixy[2];
        ait[3] = aixy[0] - aixy[3];

        yy[4] = s[3] * y[1] - s[27] * y[7];
        yy[5] = s[19] * y[5] - s[11] * y[3];
        yy[6] = c[19] * y[5] + c[11] * y[3];
        yy[7] = c[3] * y[1] + c[27] * y[7];

        y[4] = yy[4] + yy[5];
        y[5] = yy[4] - yy[5];
        y[6] = -yy[6] + yy[7];
        y[7] = yy[6] + yy[7];

        yy[4] = y[4];
        yy[7] = y[7];
        yy[5] = c[15] * (-y[5] + y[6]);
        yy[6] = c[15] * (y[5] + y[6]);
```

```
        for (jj=0; jj<4; jj++)
                y[jj] = ait[jj] +  yy[7-jj];

        for (jj=4; jj<8; jj++)
                y[jj] = ait[7-jj] - yy[jj];

        for (jj=0; jj<B_size; jj++)
                z[jj][ii] = y[jj];

}

for (ii=0; ii<B_size; ii++) {
        for (jj=0; jj<B_size; jj++)
                y[jj] = z[ii][jj];

        ait[0] = y[0];
        ait[1] = y[2];
        ait[2] = y[4];
        ait[3] = y[6];

        aixy[0] = c[15] * (ait[0] +  ait[2]);
        aixy[1] = c[15] * (ait[0] - ait[2]);
        aixy[2] = s[7] * ait[1] - s[23] * ait[3];
        aixy[3] = c[7] * ait[1] +  c[23] * ait[3];

        ait[0] = aixy[0] +  aixy[3];
        ait[1] = aixy[1] +  aixy[2];
        ait[2] = aixy[1] - aixy[2];
        ait[3] = aixy[0] - aixy[3];

        yy[4] = s[3] * y[1] - s[27] * y[7];
        yy[5] = s[19] * y[5] - s[11] * y[3];
        yy[6] = c[19] * y[5] +  c[11] * y[3];
        yy[7] = c[3] * y[1] +  c[27] * y[7];

        y[4] = yy[4] +  yy[5];
```

```
            y[5] = yy[4] - yy[5];
            y[6] = -yy[6] + yy[7];
            y[7] = yy[6] + yy[7];

            yy[4] = y[4];
            yy[7] = y[7];
            yy[5] = c[15] * (-y[5] + y[6]);
            yy[6] = c[15] * (y[5] + y[6]);

            for (jj=0; jj<4; jj++)
                  y[jj] = ait[jj] + yy[7-jj];

            for (jj=4; jj<8; jj++)
                  y[jj] = ait[7-jj] - yy[jj];

            for (jj=0; jj<B_size; jj++)
                  z[ii][jj] = y[jj] / 4.0;

      }

      for (ii=0; ii<B_size; ii++)
            for (jj=0; jj<B_size; jj++)
                  ix[ii][jj] = nint(z[ii][jj]);
}
```