picturecolordecoder

February 9, 2017

1 Program - picturecolordecoder

Program to decode a color video from files framedimc.txt, y00encc.bin, y01encc.bin, and y10encc.bin, u00enc.bin, v00enc.bin using 2-bit DCT coefficient values - Gerald Schuller, Dec. 2015 * Import relevant modules:

```
In [ ]: import cv2
        import numpy as np
        import scipy.fftpack as sft
        #import our file functions:
        from writereadbits import *
        import blockdct

        N=8
```

- Read DC values of the DCT's from file:

```
In [ ]: #indices00=readbitsfile('y00enc.bin')
        codestring00=readbinaryfile('y00encc.bin')
        indices00=codestring2data(codestring00)
        indices00=indices00+2
```

- Read AC values from file:

```
In [ ]: #indices01=readbitsfile('y01enc.bin')
        codestring01=readbinaryfile('y01encc.bin')
        indices01=codestring2data(codestring01)

        #indices10=readbitsfile('y10enc.bin')
        codestring10=readbinaryfile('y10encc.bin')
        indices10=codestring2data(codestring10)
```

- Read in DC color components:

```
In [ ]: #indices00=readbitsfile('y00enc.bin')
        codestringu00=readbinaryfile('u00enc.bin')
        indicesu00=codestring2data(codestringu00)
```

- Subtract smallest index value to obtain original value range:

```
In [ ]: indicesu00=indicesu00;

        #indices00=readbitsfile('y00enc.bin')
        codestringv00=readbinaryfile('v00enc.bin')
        indicesv00=codestring2data(codestringv00)
```

- Subtract smallest index value to obtain original value range:

```
In [ ]: indicesv00=indicesv00;
```

- Reshape back into 2-D frame with rindex rows and cindex solumns:

```
In [ ]: #load dimensions from info file:
        [r,c]=np.loadtxt('framedimc.txt')
        rindex=r/8;
        cindex=c/8;
        indices00=np.reshape(indices00,(-1,cindex))
        indices01=np.reshape(indices01,(-1,cindex))
        indices10=np.reshape(indices10,(-1,cindex))

        #color components:
        indicesu00=np.reshape(indicesu00,(-1,cindex))
        indicesv00=np.reshape(indicesv00,(-1,cindex))

        #print('De-Quantisieren')
```

- de-quantization in the decoder:

```
In [ ]: Ydct = np.zeros((r,c));
        Udct = np.zeros((r,c));
        Vdct = np.zeros((r,c));
```

- Number of bits per pixel:

```
In [ ]: bits = 2
        # resulting quantization step size for 2^bits steps:
```

- Stufen fuer unterschiedliche Ortsfrequenzen: DC Indices mit range 0…5:

```
In [ ]: quantstufeDC = 5.0/(2**bits-1)
```

- Zwei AC Koeffizienten, mit range 0.5-(-0.5)

```
In [ ]: quantstufeAC = 1.0/(2**bits-1)
```

- DC values de-quantization:

```
In [ ]: Ydct[0::N,0::N] = indices00*quantstufeDC
```

- 2 AC values de-quantization:

```
In [ ]: Ydct[0::N,1::N] = indices01*quantstufeAC
        Ydct[1::N,0::N] = indices10*quantstufeAC
        #The rest of the DCT coefficients is not transmitted and set to zero.
```

- color components de-quatization: DC Indices mit range 0...1:

```
In [ ]: quantstufeDC = 5.0/(2**bits-1)
```

- DC values de-quantization:

```
In [ ]: Udct[0::N,0::N] = indicesu00*quantstufeDC
        Vdct[0::N,0::N] = indicesv00*quantstufeDC
```

- Inverse 2D DCT:

```
In [ ]: Y = blockdct.invdct8x8(Ydct)
        U = blockdct.invdct8x8(Udct)
        V = blockdct.invdct8x8(Vdct)

        B = U + Y
        R = V + Y
        G = (Y-0.114*B-0.299*R)/0.587
```

- Schreibe die RGB Komponenten in den rekonstruierten Frame:

```
In [ ]: framerec=np.zeros((r,c,3))
        framerec[:,:,0] = B
        framerec[:,:,1] = G
        framerec[:,:,2] = R
```

- Store decoded picture in file "decodedpic.jpg":

```
In [ ]: cv2.imwrite('decodedpic.jpg', framerec*255)
```