picturecolorencoder

February 9, 2017

1 Program - picturecolorencoder

Program to capture a video from the default camera (0), compute the 2D DCT on the Y, Cr, CB component, quantize the lowest 3 coefficients of each DCT block and save them as 2 bit values in files framedimc.txt, y00encc.bin, u00enc.bin, v00enc.bin y01encc.bin, and y10encc.bin - 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
```

```
In [ ]: cap = cv2.VideoCapture(0)
```

- Get size of frame:

```
In [ ]: [retval, frame] = cap.read()
        [r,c,d]=frame.shape
        print(r,c)
```

- Store dimensions in info file:

```
In [ ]: np.savetxt('framedimc.txt', [r,c])

        print "Record to compressed files with key 'q', show until key 'q' is pressed "
```

- only kep the 3 lowest frequencies coefficients of the 8x8 DCT,

```
In [ ]: N=8

        while(True):
            # Capture frame-by-frame
            [retval, frame] = cap.read()
            """
            YUV=cv2.cvtColor(frame, cv2.COLOR_BGR2YUV)
            #frame=cv2.cvtColor(YUV, cv2.COLOR_YUV2BGR)
```

```
print "YUV.shape : ", YUV.shape
Y=YUV[:,:,0]/255.0;
U=YUV[:,:,1]/255.0;
V=YUV[:,:,2]/255.0;
#print "Y= ", Y[0,0]
#print "U= ", U[0,0]
#print "V= ", V[0,0]
"""

# /256 because the result is float values which imshow expects in range 0...1:
Y=(0.114*frame[:,:,0]+0.587*frame[:,:,1]+0.299*frame[:,:,2])/255;

#U=B-Y:
U=frame[:,:,0]/255.0-Y;

#V=R-Y:
V=frame[:,:,2]/255.0-Y;


key=cv2.waitKey(1) & 0xFF
if key== ord('c'):
    print "store frame encoded in files framedimc.txt, y00encc.bin, y01encc.bin, and

    #compute magnitude of 2D DCT of blocks of 8x8 pixels of theY component
    Ydct=blockdct.dct8x8(Y)
    Udct=blockdct.dct8x8(U)
    Vdct=blockdct.dct8x8(V)
    print np.min(np.min(Udct))

    #Quantize:
    #print('Quantisieren')
    #Ausprobieren vom Bereich:
    #DC: 0..5
    #AC: -0.5..+0.5
    #Number of bits per pixel
    bits=2

    #resulting quantization step size for 2^bits steps:
    #Stufen fuer unterschiedliche Ortsfrequenzen:
    #DC Indices mit range 0...5:
    quantstufeDC=5.0/(2**bits-1)

    #Alle DC indices (anfangen mit Position 0 und dann jeder N'te Koeffizient:
    #Kleinsten Indexwert addieren um den ganzen range des coders zu nutzen:
    indices00=np.round(Ydct[0::N,0::N]/quantstufeDC)-2

    #reshape into 1-D array:
    indices00=np.reshape(indices00,(1,-1))
```

```
#print indices00.shape

#convert to code string:
codestring00=data2codestring(indices00[0,:])

#write  to binary file
writebinaryfile('y00encc.bin', codestring00)


#Zwei AC Koeffizienten, mit range 0.5-(-0.5)
#DCT Koeffizienten der Position (0,1):
quantstufeAC=1.0/(2**bits-1)
indices01=np.round(Ydct[0::N,1::N]/quantstufeAC)

#Reshape:
indices01=np.reshape(indices01,(1,-1))
#Store with 2 bits each value:

#convert to code string:
codestring01=data2codestring(indices01[0,:])

#write to binary file
writebinaryfile('y01encc.bin', codestring01)

#DCT Koeffizienten der Position (1,0):
indices10=np.round(Ydct[1::N,0::N]/quantstufeAC)
indices10=np.reshape(indices10,(1,-1))

#convert to code string:
codestring10=data2codestring(indices10[0,:])

#write to binary file
writebinaryfile('y10encc.bin', codestring10)

#store color components DC Value:
#DC Indices mit range 0...5:
quantstufeDC=5.0/(2**bits-1)

#kleinsten Wert addieren...
indicesU00=np.round(Udct[0::N,0::N]/quantstufeDC)

#reshape into 1-D array:
indicesU00=np.reshape(indicesU00,(1,-1))
#print indices00.shape

#convert to code string:
codestringU00=data2codestring(indicesU00[0,:])
```

```
            #write  to binary file
            writebinaryfile('u00enc.bin', codestringU00)

            #Kleinsten Indexwert addieren um den ganzen range des coders zu nutzen:
            indicesV00=np.round(Vdct[0::N,0::N]/quantstufeDC)

            #reshape into 1-D array:
            indicesV00=np.reshape(indicesV00,(1,-1))
            #print indices00.shape

            #convert to code string:
            codestringV00=data2codestring(indicesV00[0,:])

            #write  to binary file
            writebinaryfile('v00enc.bin', codestringV00)

        cv2.putText(frame,"Frame Compression Demo,", (20,50), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
        cv2.putText(frame,"Press 'c' to take picture, 'q' to quit,", (20,80), cv2.FONT_HERSH
        cv2.imshow('Original Video, Y Komponente, 8bits/Pixel',frame)

        #Keep window open until key 'q' is pressed:
        if  key == ord('q'):
            break
```

- When everything done, release the capture:

```
In [ ]: cap.release()
        cv2.destroyAllWindows()
```