videorecdctblocksidctquant

February 9, 2017

1 Program - videorecdctblocksidctquant

Program to capture a video from the default camera (0), compute the 2D DCT on the Green component, take the magnitude (phase) and display it live on the screen, divide the picture into blocks of 8x8 pixels and apply a 2D DCT to each, low pass filter, and inverse transform. - Gerald Schuller, Nov. 2014, modified Dec. 2015 * Import relevant modules:

```
In [1]: import cv2
        import numpy as np
        import scipy.fftpack as sft
```

- Define the variables:

```
In [2]: #Number of bits per Y pixel
        #bits=2
        #resulting quantization step size for 2^bits steps:
        #Stufen fuer unterschiedliche Ortsfrequenzen:
        #Bereich ist nun groesser da sich die Signalenergie auf wenige Koeffizienten verteilen k
        #quantstufe=8.0/(2**bits-1)

        cap = cv2.VideoCapture(0)
```

- Get size of frame:

```
In [3]: [retval, frame] = cap.read()
        [r,c,d] = frame.shape
        print(r,c)
```

```
(480, 640)
```

- Mask to set to zero the 3/4 highest frequencies, only kep the 1/4 lowest frequencies in each direction for the 8x8 DCT, because of the DCT no longer symmetric about the center:

- Grid of 8x8 blocks:

```
In [4]: gc = np.zeros((1,c))
        gc[0,0:c:8] = np.ones(c/8)
        gr = np.zeros((r,1))
        gr[0:r:8,0] = np.ones(r/8)
        grid = np.ones((r,1)) * gc + gr * np.ones((1, c))
        #print(grid[0:9,0:9])
        N = 8
```

- Start capturing and apply the blockwise DCT on to the captured frames:

```
In [5]: while(True):
            # Capture frame-by-frame
            [retval, frame] = cap.read()
            Y=(0.114*frame[:,:,0]+0.587*frame[:,:,1]+0.299*frame[:,:,2])/255;
            cv2.imshow('Original Video, Y Komponente, 8bits/Pixel',Y)#+grid)


            #compute magnitude of 2D DCT of blocks of 8x8 pixels of the green component
            #by first reshaping the image to width 8 and applying the 1D DCT all rows, then resh
            #then transpose it, and again reshape it to width 8 and apply the 1D DCT to each rou
            #and transpose it back.
            #with norm='ortho' for "energy conservation" in the subbands and for
            #invertibiltity without factor.

            #First reshape green frame as frame with rows of width 8, (rows: order= 'C' ),
            #and apply DCT to each row of length 8 of all blocks:
            frame=np.reshape(frame[:,:,1],(-1,8), order='C')
            X=sft.dct(frame/255.0,axis=1,norm='ortho')

            #shape it back to original shape:
            X=np.reshape(X,(-1,c), order='C')

            #Shape frame with columns of hight 8 by using transposition .T:
            X=np.reshape(X.T,(-1,8), order='C')
            X=sft.dct(X,axis=1,norm='ortho')

            #shape it back to original shape:
            X=(np.reshape(X,(-1,r), order='C')).T

            #Quantize:
            #print('Quantisieren')
            #Ausprobieren vom Bereich:
            #print X[0::N,0::N]
            #DC: ca. 0..5
            #print X[0::N,1::N]
            #AC: ca. -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 Beeich 0...5:
quantstufeDC=5.0/(2**bits-1)

#Alle DC indices (anfangen mit Position 0 und dann jeder N'te Koeffizient:
indices00=np.round(X[0::N,0::N]/quantstufeDC)
#print indices00

#Zwei AC Koeffizienten, mit range 0.5-(-0.5)
quantstufeAC=1.0/(2**bits-1)
indices01=np.round(X[0::N,1::N]/quantstufeAC)
indices10=np.round(X[1::N,0::N]/quantstufeAC)


#print('De-Quantisieren')

#de-quantization in the decoder:
Xrek=np.zeros((r,c));

#DC values de-quantization:
Xrek[0::N,0::N]=indices00*quantstufeDC

#2 AC values de-quantization:
Xrek[0::N,1::N]=indices01*quantstufeAC
Xrek[1::N,0::N]=indices10*quantstufeAC
#The rest of the DCT coefficients is not transmitted and set to zero.

#Inverse 2D DCT,
#Rows:
X=np.reshape(Xrek,(-1,8), order='C')
X=sft.idct(X,axis=1,norm='ortho')

#shape it back to original shape:
X=np.reshape(X,(-1,c), order='C')

#Shape frame with columns of hight 8 (columns: order='F' convention):
X=np.reshape(X.T,(-1,8), order='C')
x=sft.idct(X,axis=1,norm='ortho')

#shape it back to original shape:
x=(np.reshape(x,(-1,r), order='C')).T

cv2.imshow('De-Quantizer und Inverse 2D DCT, 2 bits/uebertr. Koeff ', x)

#Keep window open until key 'q' is pressed:
```

```
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

- When everything done, release the capture:

```
In [6]: cap.release()
        cv2.destroyAllWindows()
```