videorecdctblocksidctquantjpgmask

February 9, 2017

1 Program - videorecdctblocksidctquantjpgmask

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 iverse transform. - Gerald Schuller, Nov. 2014, modified Dec. 2015 * Import relevant modules.

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

- Number of bits per Y pixel resulting quantization step size for 2^bits steps: Stufen fuer unterschiedliche Ortsfrequenzen:

```
In [2]: bits=4
        quantstufe1=5.0/(2**bits-1)
        bits=3
        quantstufe2=1.0/(2**bits-1)
        bits=2
        quantstufe3=0.6/(2**bits-1)
        bits=1
        quantstufe4=0.4/(2**bits-1)
        bits=0
        quantstufe5=8.0/(2**bits-0.99)
```

- vermeide div. durch 0! Zus.: 14 *bits* + 2 3 bits + 32*bits* + 41 bits fuer 64 pixel, also 0.3125 bit pro pixel!

- Quantisierungsstufen in "Maske", anti-diagonalen haben gleiche quantstufen:

```
In [3]: M = np.zeros((8, 8))
        M[0,0] = quantstufe1
        M = M + np.fliplr(np.diag([1, 1], 6)) * quantstufe2
        M = M + np.fliplr(np.diag([1, 1, 1], 5)) * quantstufe3
        M = M + np.fliplr(np.diag([1, 1, 1, 1], 4)) * quantstufe4
        M = M + np.fliplr(np.tril(np.ones((8, 8)), 3)) * quantstufe5
```

- Statt unserer selbst gemachten Quantisierungs-maske nehmen wir die JPEG Mask, fuer nirmalisierte Pixel Werte:

```
In [4]: M=np.array([[16,11,10,16,24,40,51,61],
        [12,12,14,19,26,58,60,55],
        [14,13,16,24,40,57,69,56],
        [14,17,22,29,51,87,80,62],
        [18,22,37,56,68,109,103,77],
        [24,35,55,64,81,104,113,92],
        [49,64,78,87,103,121,120,101],
        [72,92,95,98,112,100,103,99]])/64.0


        print "Quantization Mask: \n", M
```

```
Quantization Mask:
[[ 0.25      0.171875  0.15625   0.25      0.375     0.625     0.796875
   0.953125]
 [ 0.1875    0.1875    0.21875   0.296875  0.40625   0.90625   0.9375
   0.859375]
 [ 0.21875   0.203125  0.25      0.375     0.625     0.890625  1.078125
   0.875   ]
 [ 0.21875   0.265625  0.34375   0.453125  0.796875  1.359375  1.25
   0.96875 ]
 [ 0.28125   0.34375   0.578125  0.875     1.0625    1.703125  1.609375
   1.203125]
 [ 0.375     0.546875  0.859375  1.        1.265625  1.625     1.765625
   1.4375  ]
 [ 0.765625  1.        1.21875   1.359375  1.609375  1.890625  1.875
   1.578125]
 [ 1.125     1.4375    1.484375  1.53125   1.75      1.5625    1.609375
   1.546875]]
```

```
In [5]: cap = cv2.VideoCapture(0)
        #Get size of frame:
        [retval, frame] = cap.read()
        [r,c,d]=frame.shape
        print(r,c)
```

```
(480, 640)
```

- Create quantization mask of the size of the image, using kronecker product:

```
In [6]: r8 = r/8
        c8 = c/8
        Mframe = np.kron(np.ones((r8, c8)), M)
```

- 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 [7]: 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))
```

- Start capturing and process each frame.

```
In [8]: 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, 8 bit/Pixel',Y)

            #compute the 2D DCT of blocks of 8x8 pixels of the green component, of normalized fr
            X=blockdct.dct8x8(frame[:,:,1]/255.0)

            #Quantize:
            #print('Quantisieren mit Quantisierungsmaske:')
            indices=np.round(X/Mframe)
            #print indices
            #print('De-Quantisieren')

            #de-quantization in the decoder:
            Xrek=indices*Mframe

            x=blockdct.invdct8x8(Xrek);

            cv2.imshow('De-Quantizer mit Quant.-Maske und Inverse 2D DCT (0.315 bit/Pixel)', x)

            #Keep window open until key 'q' is pressed:
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
```

- When everything done, release the capture.

```
In [9]: cap.release()
        cv2.destroyAllWindows()
```