videorecdctblocks0idctdisp

March 8, 2017

1 Videorecdctblocks0idctdisp

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

- **Import relevant modules:**

```
In [1]: import cv2
        import numpy as np
        import scipy.fftpack as sft
        import warnings
        warnings.filterwarnings('ignore')
```

- **Define the variables:**

```
In [2]: cap = cv2.VideoCapture(0)

        #Get size of frame:
        [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 keep the 1/4 lowest frequencies in each direction for the 8x8 DCT, because of the DCT no longer symmetric about the center:**

- **For rows:**

```
In [3]: Mr = np.ones(8)
        Mr[(8/4.0):r] = np.zeros((3.0/4.0*8))
```

- **For columns:**

```
In [4]: Mc = Mr
```

- **Grid of 8x8 blocks:**

```
In [5]: 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))
```

- ** 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 reshape it back, then transpose it, and again reshape it to width 8 and apply the 1D DCT to each row, reshape it back, and transpose it back. with norm='ortho' for "energy conservation" in the subbands and for invertibiltity without factor.**

```
In [6]: while(True):
            # Capture frame-by-frame
            [retval, frame] = cap.read()

            cv2.imshow('Original Video, Green Component, with superimposed 8x8 grid',frame[:,:,1
            #cv2.imshow('Original Video, Gruen Komponente',frame[:,:,1])

            #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')
            #apply row filter to each row by matrix multiplication with Mr as a diagonal matrix
            X=np.dot(X,np.diag(Mr))
            #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')
            #apply column filter to each row by matrix multiplication with Mc as a diagonal matr
            X=np.dot(X,np.diag(Mc))
            #shape it back to original shape:
            X=(np.reshape(X,(-1,r), order='C')).T
            #Set to zero the 7/8 highest spacial frequencies in each direction:
            #X=X*M
            frame=np.abs(X)

            # Display the resulting frame
```

```
            cv2.imshow('2D-DCT mit Null Setzen der hoechsten Ortsfrequenzen je 8x8 Block',frame)
            #Inverse 2D DCT,
            #Rows:
            X=np.reshape(X,(-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('Inverse 2D DCT ohne die hoechsten Ortsfrequenzen', x)

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

- ** When everything done, release the capture:**

In [7]:
```
cap.release()
cv2.destroyAllWindows()
```