CV2 Examples

January 30, 2017

0.1 Program 1 - 'colormix'

Program to display the variable mix of the 3 primary colors. Outputs a window with color components(Red, Green, Blue) and with default color factors and a mix of it as well. * Import numpy and cv2.

```
In [ ]: import numpy as np
        import cv2
```

- Initialize the variables for the dimension of color to be displayed.

```
In [ ]: #Rows and columns of image:
        rows=200
        cols=250
        print(rows,cols)
        image=np.zeros((rows,cols,3))
```

- Default color factors.

```
In [ ]: b=0.5
        g=0.5
        r=0.5
```

- Set the position of the color blocks to be displayed.

```
In [ ]: bim=np.zeros((rows,cols,3))
        bim[50:150,30:130,0]=np.ones((100,100))
        gim=np.zeros((rows,cols,3))
        gim[50:150,100:200,1]=np.ones((100,100))
        rim=np.zeros((rows,cols,3))
        rim[100:200,60:160,2]=np.ones((100,100))
```

- Start capturing and display the window.

```
In [ ]: while(True):

            image=b*bim+g*gim+r*rim;
            cv2.putText(image,"Red +-: a/y, Green: s/x, Blue: d,c", (10,20), cv2.FONT_HERSHEY_SI
            v2.putText(image,"Red:"+str(r)+" Green:"+str(g)+" Blue:"+str(b)+", quit:q", (10,40),

            cv2.imshow('Color Mixture',image)
```

- The window provides controls to alter the color factors with following keyboard inputs
  - For color 'Red', 'a' & 'y' to increase and decrease the color factor respectively
  - For color 'Green', 's' & 'x' to increase and decrease the color factor respectively
  - For color 'Blue', 'd' & 'c' to increase and decrease the color factor respectively

```
In [ ]:     key=cv2.waitKey(1) & 0xFF;
            if key == ord('a'):
                r= r+0.1;
                r=np.clip(r,0,1)
            if key == ord('y'):
                r= r-0.1;
                r=np.clip(r,0,1)
            if key == ord('s'):
                g= g+0.1;
                g=np.clip(g,0,1)
            if key == ord('x'):
                g= g-0.1;
                g=np.clip(g,0,1)
            if key == ord('d'):
                b= b+0.1;
                b=np.clip(b,0,1)
            if key == ord('c'):
                b= b-0.1;
                b=np.clip(b,0,1)
            if key == ord('q'):
                break
```

- When everything done, release the capture

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

0.2 Program 2 - imagecolordisp

Program to display all possible display colors on the screen in a color triangle. * Import numpy and cv2.

```
In [ ]: import numpy as np
        import cv2
```

- Make frames of 300x300 pixels with 3 equal color components.

```
In [ ]: frame=np.zeros((300,300,3))
        frame[:,:,0]=np.ones((300,300))*0.8
        frame[:,:,1]=np.ones((300,300))*0.6
        frame[:,:,2]=np.ones((300,300))*0.2
```

```
In [ ]: frame=frame*1.0
```

- Display the resulting frame.

```
In [ ]: cv2.imshow('frame',frame)
        while(True):
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
```

- When everything done, close windows.

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

0.3 Program 3 - imagecolormixdisp

Program to display colors and color mix on the screen. * Import numpy and cv2.

```
In [ ]: import numpy as np
        import cv2
```

- Make 2 color frames of 300x300 pixels with 3 equal color components.

```
In [ ]: Farbe1=np.ones((300,300,3))
        Farbe1[:,:,0]=np.ones((300,300))*0.6
        Farbe1[:,:,1]=np.ones((300,300))*0.4
        Farbe1[:,:,2]=np.ones((300,300))*0.1

        Farbe2=np.ones((300,300,3))
        Farbe2[:,:,0]=np.ones((300,300))*0.3
        Farbe2[:,:,1]=np.ones((300,300))*0.6
        Farbe2[:,:,2]=np.ones((300,300))*0.1
```

- Mixing of colors with random factors.

```
In [ ]: Mischfarbe=0.2*Farbe1+0.8*Farbe2
```

- Display the resulting frame.

```
In [ ]: cv2.imshow('Farbe1',Farbe1)
        cv2.imshow('Farbe2',Farbe2)
        cv2.imshow('Mischfarbe',Mischfarbe)

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

- When everything done, close windows.

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

0.4 Program - 4 imagecolortriangledisp

Program to display all possible display colors on the screen in a color triangle * Import numpy and cv2

```
In [ ]: import numpy as np
        import cv2
```

- Make frames of 300x300 pixels with 3 equal color components:

```
In [ ]: frame=np.zeros((300,300,3))
```

- 300 values between 0 and 1 for the entire intensity range on a diagonal matrix:

```
In [ ]: d=np.diag(np.linspace(1,0,300))
```

- Matrix with values from 0 to 1 from bottom to top:

```
In [ ]: A = np.dot(d,np.ones((300,300)))
```

- Red component: Transposed, increasing values from 0 to 1 from left to right:

```
In [ ]: frame[:,:,2]=np.fliplr(A.T)
```

- Green component: increasing values from 0 to 1 from bottom to top:

```
In [ ]: frame[:,:,1]=A
```

- Blue component: 1-R-G:

```
In [ ]: frame[:,:,0]=np.ones((300,300))-frame[:,:,1]-frame[:,:,2]
```

- Only keep lower triangle, where the Blue component is not negative:

```
In [ ]: frame[:,:,2]=np.tril(frame[:,:,2])
        frame[:,:,1]=np.tril(frame[:,:,1])
        frame[:,:,0]=np.tril(frame[:,:,0])
```

- lower left Pixel, 1 to the right:

```
In [ ]: print('Lower left Pixel:',frame[299,0,:])
```

- upper left Pixel:

```
In [ ]: print('Upper left Pixel:',frame[0,0,:])
```

- Lower right Pixel:

```
In [ ]: print('Lower right Pixel:',frame[299,299,:])
```

- Display the resulting color triangle

```
In [ ]: cv2.imshow('Color trinagle of possible display colors',frame)

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

- When everything done, close windows.

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

0.5 Program 5 - videorecdispRGBkey

Program to capture a video from a camera and display Original and R,G,B, components live on the screen, and switch them on and off with the keys r,g,b. * Import numpy and cv2.

```
In [ ]: import numpy as np
        import cv2

        cap = cv2.VideoCapture(0)


        Ron=True
        Gon=True
        Bon=True

        while(True):
            # Capture frame-by-frame
            [ret, frame] = cap.read()
            cv2.imshow('Original',frame)

            #Null Setzen von Farb-Komponenten:
            if Ron==False:
            #Probeweise nur Farbkomponenten durch setzen von den Y-Komponenten auf einen festen
                frame[:,:,2]=np.zeros(frame[:,:,2].shape);
            if Gon==False:
                frame[:,:,1]=np.zeros(frame[:,:,1].shape);
            if Bon==False:
                frame[:,:,0]=np.zeros(frame[:,:,0].shape);
```

- Display resulting video Display text with putText(frame, text string, position, fontFace, fontScale, color, thickness)

```
In [ ]:     cv2.putText(frame,"Key r: R comp. on/off, R on="+str(Ron), (20,50), cv2.FONT_HERSHEY
            cv2.putText(frame,"Key g: G comp. on/off, G on="+str(Gon), (20,100), cv2.FONT_HERSHE
            cv2.putText(frame,"Key b: B comp. on/off, B on="+str(Bon), (20,150), cv2.FONT_HERSHE
            cv2.imshow('Einzelne Komponenten',frame)
```

- Key inputs:

    - 'r' key to toggle red component of the video.
    - 'g' key to toggle green component of the video.
    - 'b' key to toggle blue component of the video.

```
In [ ]:     key=cv2.waitKey(1) & 0xFF;
            if key == ord('r'):
                Ron= not Ron
            if key == ord('g'):
                Gon= not Gon
            if key == ord('b'):
                Bon= not Bon
            if key == ord('q'):
                break
```

- When everything done, release the capture

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

0.6 Program 6 - videorecencdecyiqkey

Program to capture a video from a camera, transform it to YIQ, transform it back, and display it live on the screen. * Import numpy and cv2

```
In [ ]: import numpy as np
        import cv2
        from numpy.linalg import inv

        cap = cv2.VideoCapture(0)
```

- YIQ transform matrix:

```
In [ ]: T=np.array([[0.299, 0.587, 0.114],
        [0.595, -0.274, -0.321],
        [0.211,-0.522, 0.311]])
```

- Inverse color transform:Inverse matrix as array:

```
In [ ]: Tinv=inv(T)

        Yon=True
        Ion=True
        Qon=True

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

```
    framerec=np.zeros(frame.shape);

    # Display the original frame
    cv2.imshow('Original',frame)
```

- Our operations on the frames come here###### **Encoder** #######Forwaerts Farb-Transformation im Encoder:Berechnung der Luminanz-Komponente Y und der Farb-Komponenten U und V:/256 because the result is float values which imshow expects in range 0...1:np.dot applies the matrix multiplication to the last axis, hence here the RGB axis:

Result is 3-dimensional matrix, rows x columns x color components

```
In [ ]:     YIQ= np.dot(frame,T)/255.0
```

- ###### **Decoder** ######Inverse Farb-Transformation im Decoder:

```
In [ ]:     if Yon==False:
              #Probeweise nur Farbkomponenten durch setzen von den Y-Komponenten auf einen feste
              YIQ[:,:,0]=np.ones(YIQ[:,:,0].shape)*0.5;
            #Probeweise Null setzen von Farb-Komponenten:
            if Ion==False:
                YIQ[:,:,1]=np.zeros(YIQ[:,:,1].shape);
            if Qon==False:
                YIQ[:,:,2]=np.zeros(YIQ[:,:,2].shape);
```

- Inverse color transform: np.dot applies the matrix multiplication to the last axis, hence here the color axis:

```
In [ ]: framerec=np.dot(YIQ,Tinv)
```

- Display reconstructed videoDisplay text with:

```
In [ ]:     cv2.putText(framerec,"Key y: Y comp. on/off, Y on="+str(Yon), (20,50), cv2.FONT_HERS
            cv2.putText(framerec,"Key i: I comp. on/off, I on="+str(Ion), (20,100), cv2.FONT_HER
            cv2.putText(framerec,"Key q: Q comp. on/off, Q on="+str(Qon), (20,150), cv2.FONT_HER

            cv2.imshow('Reconstructed, exit with x',framerec)
```

- Key inputs:
  - 'y' key to toggle Y component of the video.
  - 'i' key to toggle I component of the video.
  - 'q' key to toggle Q component of the video.
  - 'x' key to exit.

```
In [ ]:     key=cv2.waitKey(1) & 0xFF;
            if key == ord('y'):
                Yon= not Yon
            if key == ord('i'):
                Ion= not Ion
            if key == ord('q'):
                Qon= not Qon
            #Ende durch Taste "x":
            if key == ord('x'):
                break
```

- When everything done, release the capture.

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

0.7 Program 7 - videorecencdecyuvkey

Program to capture a video from a camera, transform it to YUV, transform it back, and display it live on the screen * Import numpy and cv2

```
In [ ]: import numpy as np
        import cv2

        while(True):
            # Capture frame-by-frame
            [ret, frame] = cap.read()
            framerec=np.zeros(frame.shape);

            # Display the original frame
            cv2.imshow('Original',frame)
```

- Our operations on the frames come here ###### **Encoder** ####### Forwaerts Farb-Transformation im Encoder: Berechnung der Luminanz-Komponente Y und der Farb-Komponenten U und V: Y= 0.114*B+0.587*G+0.299*R : /256 because the result is float values which imshow expects in range 0...1:

```
In [ ]:     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;
```

- #######**Decoder** #######Inverse Farb-Transformation im Decoder:

```
In [ ]:     if Yon==False:
              #Probeweise nur Farbkomponenten durch setzen von den Y-Komponenten auf einen feste
              Y=np.ones(Y.shape)*0.5;
            #Probeweise Null setzen von Farb-Komponenten:
            if Uon==False:
                U=np.zeros(U.shape);
            if Von==False:
```

```
    V=np.zeros(V.shape);

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

- Write the RGB components in the reconstruction frame.

In [ ]:
```
framerec[:,:,0]=B
framerec[:,:,1]=G
framerec[:,:,2]=R
```

- Display reconstructed video Display text with:

In [ ]:
```
cv2.putText(framerec,"Key y: Y comp. on/off, Y on="+str(Yon), (20,50), cv2.FONT_HERS
cv2.putText(framerec,"Key u: U comp. on/off, U on="+str(Uon), (20,100), cv2.FONT_HER
cv2.putText(framerec,"Key v: V comp. on/off, V on="+str(Von), (20,150), cv2.FONT_HER

cv2.imshow('Reconstructed, exit with q',framerec)
```

- Key inputs:
    - 'y' key to toggle Y component of the video.
    - 'u' key to toggle U component of the video.
    - 'v' key to toggle V component of the video.
    - 'q' key to exit.

In [ ]:
```
key=cv2.waitKey(1) & 0xFF;
if key == ord('y'):
    Yon= not Yon
if key == ord('u'):
    Uon= not Uon
if key == ord('v'):
    Von= not Von
#Ende durch Taste "q":
if key == ord('q'):
    break
```

- When everything done, release the capture

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

0.8 Program 8 - videorecprocyuv

Program to capture a video from a camera and display it live on the screen * Import numpy and cv2.

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

        cv2.namedWindow('Original')
        cv2.namedWindow('Luminanz Y')
        cv2.namedWindow('Farbkomponente U')
        cv2.namedWindow('Farbkomponente V')
        while(True):
            # Capture frame-by-frame
            [ret, frame] = cap.read()
```

- Our operations on the frames come here Berechnung der Luminanz-Komponente Y: Y= 0.114*B+0.587*G+0.299*R : /256 because the result is float values which imshow expects in range 0...1:

```
In [ ]:     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;
```

- Display the resulting frame

```
In [ ]:     cv2.imshow('Original',frame)
            cv2.imshow('Luminanz Y',Y)
            cv2.imshow('Farbkomponente U',np.abs(U))
            cv2.imshow('Farbkomponente V',np.abs(V))
```

- End by pressing the key ’q’.

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

- When everything done, release the capture

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