Electronic Supplementary Material (ESI) for Analytical Methods.

Supplementary Information

The code used to control the prototype and acquire de average of HSV written in python is presented here:

```
import cv2 as cv
import numpy as np
from gpiozero import LED

k = cv.waitKey(1)
cam1 = cv.VideoCapture(2) #Lateral Camera
cam1.set(cv.CAP_PROP_AUTO_EXPOSURE,0.25)#Automatic exposure in 0.25 =OFF and
0.75 = ON
cam1.set(cv.CAP_PROP_EXPOSURE, 0.1)# Exposure in 0.1 is the standard in the
program
cam1.set(cv.CAP_PROP_AUTOFOCUS, 0.25)# Auto Focus in 0.25 = OFF and 0.75 = ON
cam1.set(cv.CAP_PROP_FOCUS,1)# Focus in 1 to be the nearest possible
cam2 = cv.VideoCapture(0)# Frontal Camera
cam2.set(cv.CAP_PROP_AUTO_EXPOSURE,0.25)# As lateral
cam2.set(cv.CAP_PROP_EXPOSURE, 0.1)# As lateral
cam2.set(cv.CAP_PROP_AUTOFOCUS, 0.25)# As lateral
cam2.set(cv.CAP_PROP_FOCUS,1)# As lateral


Xmaxe = 470 # Max custom X coordinate
Xmine = 131 # Min custom X coordinate
Ymaxe = 298 # Max custom Y coordinate
Ymine = 172 # Min custom Y coordinate
i = 0 #crosshair variations (0 to 4)
F = LED(2)

DB = open(r'database','a+')


def RGB_pxValue(Yp,Xp):
    print("frontal")
    R = framer1[Yp,Xp,2]
    G = framer1[Yp,Xp,1]
    B = framer1[Yp,Xp,0]
    print((R,G,B))
    print("lateral")
    R = framer2[Yp,Xp,2]
    G = framer2[Yp,Xp,1]
    B = framer2[Yp,Xp,0]
    print ((R,G,B))
```

```
def RGB_Values_Retangulo(Ymaxf,Yminf,Xmaxf,Xminf,Ymaxl,Yminl,Xmaxl,Xminl):
#Scanning function and RGB to HSV translation
   Hm = 0
   Sm = 0
   Vm = 0
   V = []
   Vmed = []
   for y in range(Yminl,(Ymaxl+1)):
      for x in range(Xminl,(Xmaxl+1)):
         R = framer1[y,x,2]
         G = framer1[y,x,1]
         B = framer1[y,x,0]
         v = np.uint8([[[B,G,R]]])
         v = cv.cvtColor(v,cv.COLOR_BGR2HSV)
         V += [(((v[0][0][0])*2),int((v[0][0][1])*100/255),int((v[0][0][2])*100/255))]
         Hm += v[0][0][0]
         Sm += v[0][0][1]
         Vm += v[0][0][2]
   Vmed = (int((Hm/len(V))*2),int((Sm/len(V))*100/255),int((Vm/len(V))*100/255))
   DB.write(str(Vmed))
   print ("Scanning complete")
   print ("Mean HSV Value:")
   print (Vmed)

   for y in range(Yminf,(Ymaxf+1)):
      for x in range(Xminf,(Xmaxf+1)):
         R = framer2[y,x,2]
         G = framer2[y,x,1]
         B = framer2[y,x,0]
         v = np.uint8([[[B,G,R]]])
         v = cv.cvtColor(v,cv.COLOR_BGR2HSV)
         V += [(((v[0][0][0])*2),int((v[0][0][1])*100/255),int((v[0][0][2])*100/255))]
         Hm += v[0][0][0]
         Sm += v[0][0][1]
         Vm += v[0][0][2]
   Vmed = (int((Hm/len(V))*2),int((Sm/len(V))*100/255),int((Vm/len(V))*100/255))
   DB.write(',' + str(Vmed)+'\n')
   print ("Mean HSV Value:”)
   print (Vmed)
   print ("Press ESC to leave or SPACE to make another scan")


def sight(): #Central Crosshair
   cv.line(framev1,(310,240),(330,240),(255,0,0),1)
   cv.line(framev1,(320,230),(320,250),(255,0,0),1)
   cv.circle(framev1,(320,240),2,(255,0,0), -1)
   cv.circle(framev1,(320,240),10,(255,0,0), 1)
```

```
    cv.line(framev2,(310,240),(330,240),(255,0,0),1)
    cv.line(framev2,(320,230),(320,250),(255,0,0),1)
    cv.circle(framev2,(320,240),2,(255,0,0), -1)
    cv.circle(framev2,(320,240),10,(255,0,0), 1)
    if k%256 == 32:
        #SPACE pressed
        RGB_pxValue(240,320)


def area_default(): #Standard work area
    cv.rectangle(framev1,(484,326),(133,158),(30,255,80),2)
    cv.rectangle(framev2,(519,268),(96,168),(30,255,80),2)
    if k%256 == 32:
        #SPACE pressed
        RGB_Values_Retangulo(326,158,484,133,268,168,519,96)


def area_custom(): #Custom work area
    cv.rectangle(framev1,(Xmaxe,Ymaxe),(Xmine,Ymine),(30,255,80),2)
    cv.rectangle(framev2,(Xmaxe,Ymaxe),(Xmine,Ymine),(30,255,80),2)
    if k%256 == 32:
        #SPACE pressed
        RGB_Values_Retangulo(Ymaxe,Ymine,Xmaxe,Xmine,Ymaxe,Ymine,Xmaxe,Xmine)
        print(Xmaxe, Xmine, Ymaxe, Ymine)
    settings()

def settings(): #Custom work area adjustments
    global Xmaxe
    global Xmine
    global Ymaxe
    global Ymine

    if k%256 == 118:
        #V pressed
        print("Xmaxe = "+str(Xmaxe)+"\nXmine = "+str(Xmine)+"\nYmaxe =
"+str(Ymaxe)+"\nYmine = "+str(Ymine))

    if k%256 == 100:
        #Depressed
        #Wider
        Xmaxe += 1
        Xmine -= 1
    if k%256 == 97:
        #A pressed
        #Stretching
        Xmaxe -= 1
```

```
            Xmine += 1
        if k%256 == 119:
            #W pressed
            #Taller
            Ymaxe += 1
            Ymine -= 1
        if k%256 == 115:
            #S pressed
            #Shorter
            Ymaxe -= 1
            Ymine += 1
        if k%256 == 113:
            #Q pressed
            #To the left
            Xmaxe -= 1
            Xmine -= 1
        if k%256 == 101:
            #E pressed
            #To the right
            Xmaxe += 1
            Xmine += 1
        if k%256 == 114:
            #R pressed
            # Moves Up
            Ymaxe -= 1
            Ymine -= 1
        if k%256 == 102:
            #F pressed to pay respect
            #Moves Down
            Ymaxe += 1
            Ymine += 1
        if k%256 == 122:
            #Z pressed
            #Goes back to inicial form
            Xmaxe = 379
            Xmine = 200
            Ymaxe = 424
            Ymine = 60

    def database():# in construction
        brand = input('beer brand: ')
        with open('database', 'r') as f:
            in_file = f.readlines
        if brand == '1':

            return 0
```

```
def luminosity():
    if l == 0:
        f.on()
    if l == 1:
        f.off()


def Modes():# Scan modes
    if i == 0:
        # Initial centered sight
        sight()
    elif i == 1:
        # Standard centered rectangle
        area_default()
    elif i == 2:
        # Editable centered rectangle
        area_custom()


def Keyboard():# Primary keys
    global i, e, l
    if k%256 == 49:
        print("-----------------------------------------------" + "\n|||Crosshair Mode|||" + "\n-
Press SPACE for a central RGB value")
        # 1 pressed
        i = 0
    elif k%256 == 50:
        # 2 pressed
        print("-----------------------------------------------" + "\n|||Standard Scan Mode|||" +
"\nPress SPACE for a mean HSV Scan ")
        DB.write('-------------------------\n' + input('What is being analyzed: ')+'\n' +
r'#Openned in ' + input('Openning date: ') + ' | ' + 'Batch: ' + input('Input Batch: ') + '\n')
        i = 1

    elif k%256 == 51:
        # 3 pressed
        print("-----------------------------------------------" + "\n||| Editable scan mode |||"+"\n-
press W or S for vertical resizing"+"\n-press A or D for horizontal resizing"+"\n-press Q
or E for horizontal movement"+"\n-press R or F for vertical movement"+"\n-press Z to
restart the editable area "+"\npress ESPAÇO to collect HSV values ")
        i = 2
    elif k%256 == 9:
        #TAB pressed
        print (l)
        l = l + 1
```

```
        if l == 2:
            l = 0
    elif k%256 == 99:
        #C pressed
        DB.write('-----------------------\n' + input(' Write what is being analyzed: ') + '\n'+
r'#Open in' + input('Enter the date it was opened: ') + ' | ' + 'Lot: ' + input(' Enter the
lot: ') + '\n')

print ("1 - CrossHair mode" + "\n2 - Default mode" + "\n3 - editable mode" + "\nESC to
quit")
while True: # Image cycle for preview
    ret, framer1 = cam1.read()#framer = real frame for operation
    ret, framev1 = cam1.read()#framev = visualized frame not mirrored
    ret, framer2 = cam2.read()#idem framer1
    ret, framev2 = cam2.read()#idem framev1
    k = cv.waitKey(1)

    if not ret:
        break

    if k%256 == 27:
        #ESC pressed
        break

    luminosity()
    Keyboard()
    Modes()

    cv.imshow("side_view", framev2)
    cv.imshow("front_view", framev1)


cam1.release()
cv.destroyAllWindows()
DB.close()
exit()
```