Supporting Information

Luminescent Carbon Dot Mimics Assembled on DNA

Ke Min Chan, Wang Xu, Hyukin Kwon, Anna M. Kietrys and Eric T. Kool*

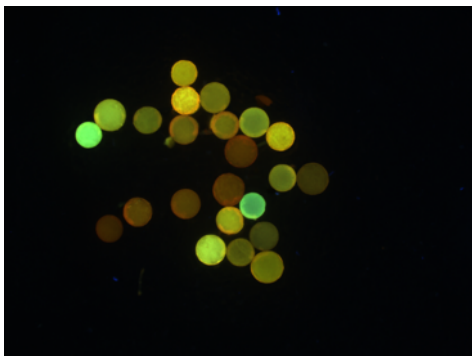
Department of Chemistry, Stanford University, Stanford, CA 94305 (USA)

Email: kool@stanford.edu

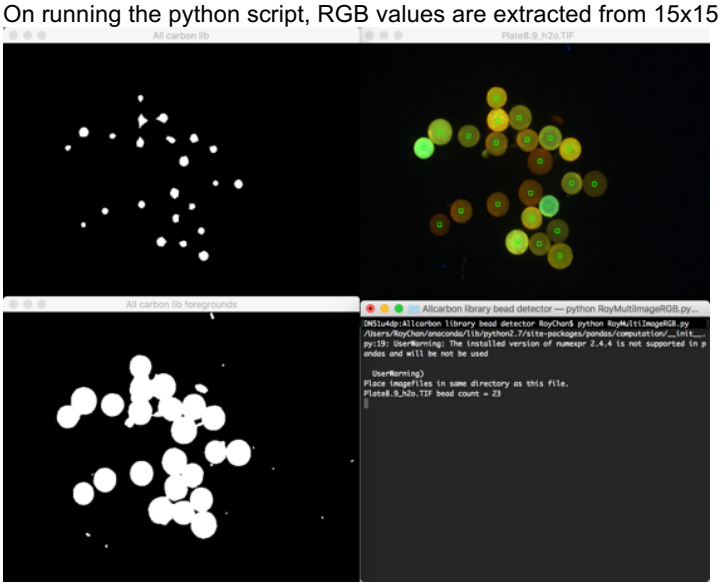
*Author to whom correspondence should be addressed.

Source code for DNA-CA library bead analysis program

Bead detector script for high throughput screening of 130 micrometer library beads. Written in Python, requires Numpy, OpenCV 2.4.11, and Pandas packages. Input file is a 1392 × 1040 image containing non-overlapping fluorescent beads against a black background:



On running the python script, RGB values are extracted from 15x15 pixel region at the center of each bead:



Sample output is a .csv file containing RGB values for each detected bead:

	A	B	C	D	E	F	G	H	I
1	Hue	Saturation	Value	R	G	B	luminosity	image_name	
2	67.3848768	0.90436431	0.39059624	88.5153061	99.6020408	9.5255102	90.4232783	Plate8.9_h2o.TIF	
3	64.0800186	0.8461488	0.30496198	73.2908163	77.7653061	11.9642857	71.9184188	Plate8.9_h2o.TIF	
4	81.2829243	0.72782531	0.85582233	161.892857	218.234694	59.3979592	190.251074	Plate8.9_h2o.TIF	
5	68.1721948	0.75257967	0.26370548	60.3520408	67.244898	16.6377551	61.4407566	Plate8.9_h2o.TIF	
6	38.278032	0.93961656	0.22332933	56.9489796	37.5765306	3.43877551	42.4252544	Plate8.9_h2o.TIF	
7	67.6577657	0.86323151	0.63797519	144.760204	162.683673	22.25	147.843382	Plate8.9_h2o.TIF	

```python
import sys
import numpy as np
import cv2
import copy
import pandas
import os
import csv

def writecsv(data):
    with open('output.csv', 'w') as fp:
        a = csv.writer(fp, delimiter=',')
        a.writerows(data)

def rgbtohsv(r,g,b):
    # rgb to hue
    r, g, b = r/255.0, g/255.0, b/255.0
    mx = max(r, g, b)
    mn = min(r, g, b)
    df = mx-mn
    if mx == mn:
        h = 0
    elif mx == r:
        h = (60 * ((g-b)/df) + 360) % 360
    elif mx == g:
        h = (60 * ((b-r)/df) + 120) % 360
    elif mx == b:
        h = (60 * ((r-g)/df) + 240) % 360
    if mx == 0:
        s = 0
    else:
        s = df/mx
    v = mx
    return h, s, v

def processImage(imgFilename,data):
    MIN_BEAD_AREA = 5
    CUT_OFF_AREA = 1500
    DARKNESS_THRESHOLD = 20 #out of 255

    #get image by filename
    img = cv2.imread(imgFilename)
    img2 = copy.copy(img)

    # Find size of img
    sz = img.shape

    #get grayscale img
```

```
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img_gray = img_gray

    #maximum among the 3 color channels
    img_max = np.amax(img,axis=2)
    img_blue = img[:,:,0]
    img_green = img[:,:,1]
    img_red = img[:,:,2]

    #picks out bead centers for image#
    ret, thresh = cv2.threshold(img_max,DARKNESS_THRESHOLD,255,cv2.THRESH_BINARY)
    #ret, thresh = cv2.threshold(img_max,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

    #noise removal
    kernel = np.ones((3,3),np.uint8)
    noiselessimg = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)

    # Finding sure foreground area
    dist_transform = cv2.distanceTransform(noiselessimg,2,5)
    ret, sure_fg = cv2.threshold(dist_transform,0.65*dist_transform.max(),255,0)

    # Finding unknown region
    sure_fg = np.uint8(sure_fg)
    sure_fg_copy = copy.deepcopy(sure_fg)

    #get contours
    contours, hierarchy = cv2.findContours( sure_fg_copy,cv2.RETR_LIST, \
                            cv2.CHAIN_APPROX_SIMPLE)

    #process individual contours
    allColors = []
    hist = []
    beadcount = 0
    for i, contour in enumerate(contours):
        if cv2.contourArea(contour) > MIN_BEAD_AREA :
            area = cv2.contourArea(contour)

            if area > CUT_OFF_AREA:
                #For overlapping beads
                x,y,w,h = cv2.boundingRect(contour)
                cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
                #do nothing elses

            else:
             M = cv2.moments(contour)
             #calculate centroid
             cx = int(M['m10']/M['m00'])
```

```
            cy = int(M['m01']/M['m00'])

            PADDING = 30

            #check if bead is not at edge of image
            if cx > PADDING and cx < (img.shape[1]-PADDING) and \
            cy > PADDING and cy< (img.shape[0]-PADDING):
                cropsquarecoords = [cx,cy]
                cv2.rectangle(img, (cx-7, cy-7), (cx+7, cy+7), (0, 255, 0), 2)

                imgcropsquare = img2[(cy -7 ):(cy + 7),(cx -7 ):(cx + 7) ,:]

                mean = cv2.mean(imgcropsquare) #[3,]

                rgb = [mean[2],mean[1],mean[0]]
                #convert rgb to hue
                h,s,v = rgbtohsv(mean[2],mean[1],mean[0])

                # HSP Color Model
                #sqrt( 0.299*R^2 + 0.587*G^2 + 0.114*B^2 )
                l = np.sqrt(0.299*mean[2]**2 + 0.587*mean[1]**2 \
                +  0.114*mean[0]**2)
                allColors.append([h,rgb,l,cropsquarecoords])

                #Hue,Saturation,Value, R, G, B, luminosity, image_name
                data.append([h,s,v,mean[2],mean[1],mean[0],l,imgFilename])

                beadcount += 1

    if beadcount == 0:
        print "beadcount in %s is 0 !" % imgFilename

    else:
        print "%s bead count = %i " % (imgFilename ,beadcount)
        colorsDataFrame = pandas.DataFrame(allColors)
        colorsDataFrame.columns = ['hue','rgb','luminance', 'coords']
        sortedDF = colorsDataFrame.sort_values(by=['hue'],ascending=[True])

        #show image
        resizedfgcopy = cv2.resize(dist_transform,(500,400))
        cv2.imshow('All carbon lib foregrounds',resizedfgcopy)

        cv2.moveWindow('All carbon lib foregrounds',0,400)
        resizedimg = cv2.resize(sure_fg,(500,400))
        cv2.imshow('All carbon lib',resizedimg)
        resizedimg = cv2.resize(img,(500,400))
        cv2.imshow(imgFilename,resizedimg)
```

```
        cv2.moveWindow(imgFilename,500,0)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

print "Place imagefiles in same directory as this file."
path = "./"
dirs = os.listdir(path)

data = []
data.append(["Hue","Saturation","Value", "R", "G", "B", "luminosity",\
     "image_name"])

for file in dirs:
    if file.endswith('.TIF'):
        processImage(file,data)

writecsv(data)
```