



import numpy as np

import matplotlib.pyplot as plt

import argparse

import cv2

parser = argparse.ArgumentParser()

parser.add\_argument('-f', '--file',

help='Path to video file (if not using camera)')

parser.add\_argument('-c', '--color', type=str, default='gray',

help='Color space: "gray" (default), "rgb", or "lab"')

parser.add\_argument('-b', '--bins', type=int, default=16,

help='Number of bins per channel (default 16)')

parser.add\_argument('-w', '--width', type=int, default=0,

help='Resize video to specified width in pixels (maintains aspect)')

args = vars(parser.parse\_args())

# Configure VideoCapture class instance for using camera or file input.

if not args.get('file', False):

capture = cv2.VideoCapture(0)

else:

capture = cv2.VideoCapture(args['file'])

color = args['color']

bins = args['bins']

resizeWidth = args['width']

# Initialize plot.

fig, ax = plt.subplots()

if color == 'rgb':

ax.set\_title('Histogram (RGB)')

elif color == 'lab':

ax.set\_title('Histogram (L\*a\*b\*)')

else:

ax.set\_title('Histogram (grayscale)')

ax.set\_xlabel('Bin')

ax.set\_ylabel('Frequency')

# Initialize plot line object(s). Turn on interactive plotting and show plot.

lw = 3

alpha = 0.5

if color == 'rgb':

lineR, = ax.plot(np.arange(bins), np.zeros((bins,)), c='r', lw=lw, alpha=alpha, label='Red')

lineG, = ax.plot(np.arange(bins), np.zeros((bins,)), c='g', lw=lw, alpha=alpha, label='Green')

lineB, = ax.plot(np.arange(bins), np.zeros((bins,)), c='b', lw=lw, alpha=alpha, label='Blue')

elif color == 'lab':

lineL, = ax.plot(np.arange(bins), np.zeros((bins,)), c='k', lw=lw, alpha=alpha, label='L\*')

lineA, = ax.plot(np.arange(bins), np.zeros((bins,)), c='b', lw=lw, alpha=alpha, label='a\*')

lineB, = ax.plot(np.arange(bins), np.zeros((bins,)), c='y', lw=lw, alpha=alpha, label='b\*')

else:

lineGray, = ax.plot(np.arange(bins), np.zeros((bins,1)), c='k', lw=lw, label='intensity')

ax.set\_xlim(0, bins-1)

ax.set\_ylim(0, 1)

ax.legend()

plt.ion()

plt.show()

# Grab, process, and display video frames. Update plot line object(s).

while True:

(grabbed, frame) = capture.read()

if not grabbed:

break

# Resize frame to width, if specified.

if resizeWidth > 0:

(height, width) = frame.shape[:2]

resizeHeight = int(float(resizeWidth / width) \* height)

frame = cv2.resize(frame, (resizeWidth, resizeHeight),

interpolation=cv2.INTER\_AREA)

# Normalize histograms based on number of pixels per frame.

numPixels = np.prod(frame.shape[:2])

if color == 'rgb':

cv2.imshow('RGB', frame)

(b, g, r) = cv2.split(frame)

histogramR = cv2.calcHist([r], [0], None, [bins], [0, 255]) / numPixels

histogramG = cv2.calcHist([g], [0], None, [bins], [0, 255]) / numPixels

histogramB = cv2.calcHist([b], [0], None, [bins], [0, 255]) / numPixels

lineR.set\_ydata(histogramR)

lineG.set\_ydata(histogramG)

lineB.set\_ydata(histogramB)

elif color == 'lab':

cv2.imshow('L\*a\*b\*', frame)

lab = cv2.cvtColor(frame, cv2.COLOR\_BGR2LAB)

(l, a, b) = cv2.split(lab)

histogramL = cv2.calcHist([l], [0], None, [bins], [0, 255]) / numPixels

histogramA = cv2.calcHist([a], [0], None, [bins], [0, 255]) / numPixels

histogramB = cv2.calcHist([b], [0], None, [bins], [0, 255]) / numPixels

lineL.set\_ydata(histogramL)

lineA.set\_ydata(histogramA)

lineB.set\_ydata(histogramB)

else:

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

cv2.imshow('Grayscale', gray)

histogram = cv2.calcHist([gray], [0], None, [bins], [0, 255]) / numPixels

lineGray.set\_ydata(histogram)

fig.canvas.draw()

if cv2.waitKey(1) & 0xFF == ord('q'):

break

capture.release()

cv2.destroyAllWindows()