REPORT

Melih Kağan Özçelik

PART1

I just added a while loop to iterate through cat frames on /cat folder.

```
imageslist = []
i = 0
while 1: # loop through cat frames
    image=cv2.imread('cat/cat_'+ str(i) + '.png')
    if image is None:
        break
    i = i+1
   ……
   ……
```

PART 2

To obtain mirror image of cat , I flip the cat image and merge it with rectengular raw green image side by side.

```
#place the reflected cat to the right side
mirror_im = cv2.flip(image,1)
green_im = np.zeros((360,286,3), np.uint8)
green_im[:,0:286]= (0,255,0)
mirror_im = cv2.hconcat([green_im, mirror_im])
```

And then I changed the empty lef side of mirror_im with the pixel values of original cat image.

```
with_myself = mirror_im.copy()
with_myself[0:image.shape[0],0:image.shape[1]-110,:] =
image[0:image.shape[0],0:image.shape[1]-110,:]
#i left 110 more pixels to right cat to see its head shaking better , otherwise it was cropping by orginal cat figures
```

PART 3

To make the cat on the right darker, I first tried to reach every pixel value and change it however I could not even wait the results because this way takes too much time.

```
g = 25
darker_cat = np.zeros(mirror_im.shape, mirror_im.dtype)
for y in range(mirror_im.shape[0]):
    for x in range(mirror_im.shape[1]):
        for c in range(mirror_im.shape[2]):
            if c==1 and mirror_im[y,x,c] >= 235: #to keep green screen
                darker_cat[y,x,c] = mirror_im[y,x,c]
            else:
                if mirror_im[y,x,c] - g < 0 :
                    darker_cat[y,x,c] = 0
                else:
                    darker_cat[y,x,c] = mirror_im[y,x,c] - g
```

Then , I tried the method called vectorization.

```
def darker(pix,g):
    if (pix-g) <0 :
        return 0
    else:
        return (pix-g)
```

```
g=90 # paremater for brightness
darker_vec = np.vectorize(darker)
…
   darker_cat_g = darker_g_vec(mirror_g_channel,g)
   darker_cat_r = darker_vec(mirror_r_channel,g)
   darker_cat_b = darker_vec(mirror_b_channel,g)


   darker_cat = np.dstack((darker_cat_r,darker_cat_g,darker_cat_b))
```

Then I encountered some pixel problems :

To solve that problem I used the only reflected cats values on the frames, not the green background :

```
  #only use darker cat values in frames
   with_myself = mirror_im.copy()
   foreground = np.logical_or(mirror_g_channel < 180 , mirror_r_channel > 150)
   n_x,n_y = np.nonzero(foreground)
   darker_cat_values = darker_cat[n_x, n_y, :]
   with_myself[n_x,n_y, : ] = darker_cat_values
   #and the left part, bright cat
   with_myself[0:image.shape[0],0:image.shape[1]-110,:] =
image[0:image.shape[0],0:image.shape[1]-110,:]
```

PART 4

Target image (yellow.jpg) :

Result:

Target image (blue.jpg):

Result:

To calculate average cat histogram , I got every frames histogram in a loop and added them to the list , then I took the average of lists for every channel.

```
b_h_list = []
g_h_list = []
r_h_list = []
i = 0
while 1:
……
        mirror_hist_blue, bin_0 = np.histogram(mirror_b_channel.flatten(), 256, [0,256])
         mirror_hist_green, bin_1 = np.histogram(mirror_g_channel.flatten(), 256, [0,256])
        mirror_hist_red, bin_2 = np.histogram(mirror_r_channel.flatten(), 256, [0,256])
        b_h_list.append(mirror_hist_blue)
        g_h_list.append(mirror_hist_green)
         r_h_list.append(mirror_hist_red)
……
total_b = np.zeros_like(mirror_hist_blue)
for a in b_h_list:
   total_b = total_b + a
average_b_histogram = total_b/i    #i = 180 in here
…..
```

Then I created lookup table using CDF's of both target images channels and cat images channels.

```
source_cdf_blue = calculate_cdf(average_b_histogram)
source_cdf_green = calculate_cdf(average_g_histogram)
source_cdf_red = calculate_cdf(average_r_histogram)
target_cdf_blue = calculate_cdf(target_hist_blue)
target_cdf_green = calculate_cdf(target_hist_green)
target_cdf_red = calculate_cdf(target_hist_red)

blue_lookup_table = lookup(source_cdf_blue, target_cdf_blue)
green_lookup_table = lookup(source_cdf_green, target_cdf_green)
red_lookup_table = lookup(source_cdf_red, target_cdf_red)
```

Finally , for every frame I applied the transform using LUT:

```
b_after_transform = cv2.LUT(mirror_b_channel, blue_lookup_table)
 g_after_transform = cv2.LUT(mirror_g_channel, green_lookup_table)
 r_after_transform = cv2.LUT(mirror_r_channel, red_lookup_table)

 image_after_matching = cv2.merge([b_after_transform, g_after_transform,
r_after_transform])
  image_after_matching = cv2.convertScaleAbs(image_after_matching)
```