get most dominant colors from video - openCV PYTHON

Asked 2 months ago Modified 2 months ago Viewed 72 times

0

im trying to get the most dominant colors from a video, when starting to play want to draw in real time the colors from the video, for example the 6 most dominant, 3, etc, i searched a lot but all tutorials they all detecting only three colors, red, blue and green, someone may be detecting a bit more because they are setting the values by themselves, using the hsv map to they can set which colors detect, mine problem is that is a video, so i wont know the ranges

```
while(True):

    # Capture the video frame
    # by frame
    ret, frame = vid.read();
    prev = time.time();

    capture = cv.VideoCapture(args['file'])
    img = cv.imread("./assets/taxi.jpeg");

    rgb_color = cv.cvtColor(frame, cv.COLOR_BGR2RGB);
    height, width, channel = rgb_color.shape;

    histogram = cv.calcHist([frame],[0],None,[256],[0,256]);
    plt.plot(histogram);
    cv.imshow("histogram", plt);
```

for now just open the webcam and showing the histogram

python opencv

Share Follow

asked Sep 22 at 2:56
plus
299 3 10

Reduce the number of colors in your image using kmeans. – fmw42 Sep 22 at 4:54

when using kmeans inside the while loop its too slow – plus Sep 22 at 5:07

cv.imshow("histogram", plt) to cv.imshow("histogram", histogram) – toyota Supra Sep 22 at 8:59

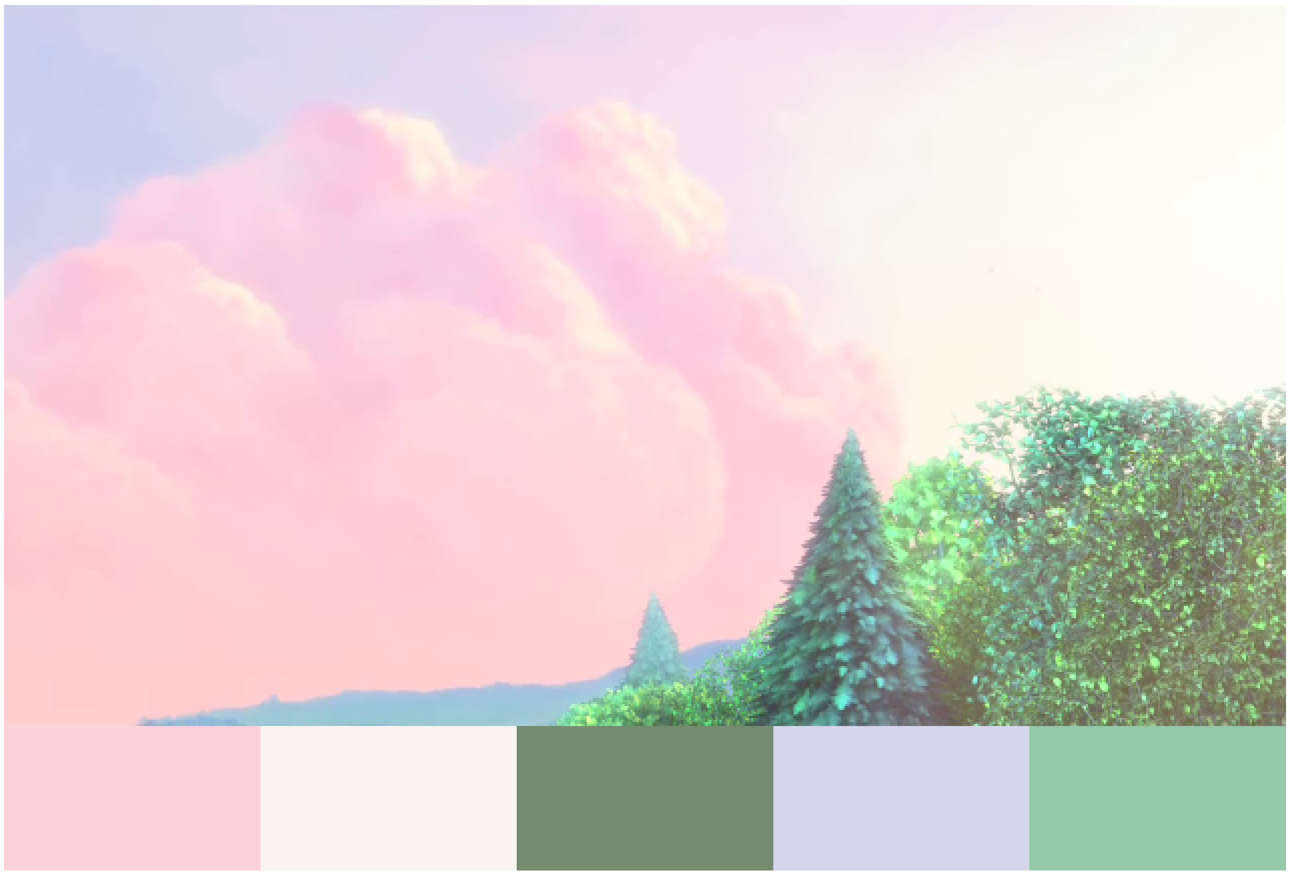
1 Answer

Sorted by: Highest score (default)

Join Stack Overflow to find the best answer to your technical question, help others answer theirs. Sign up

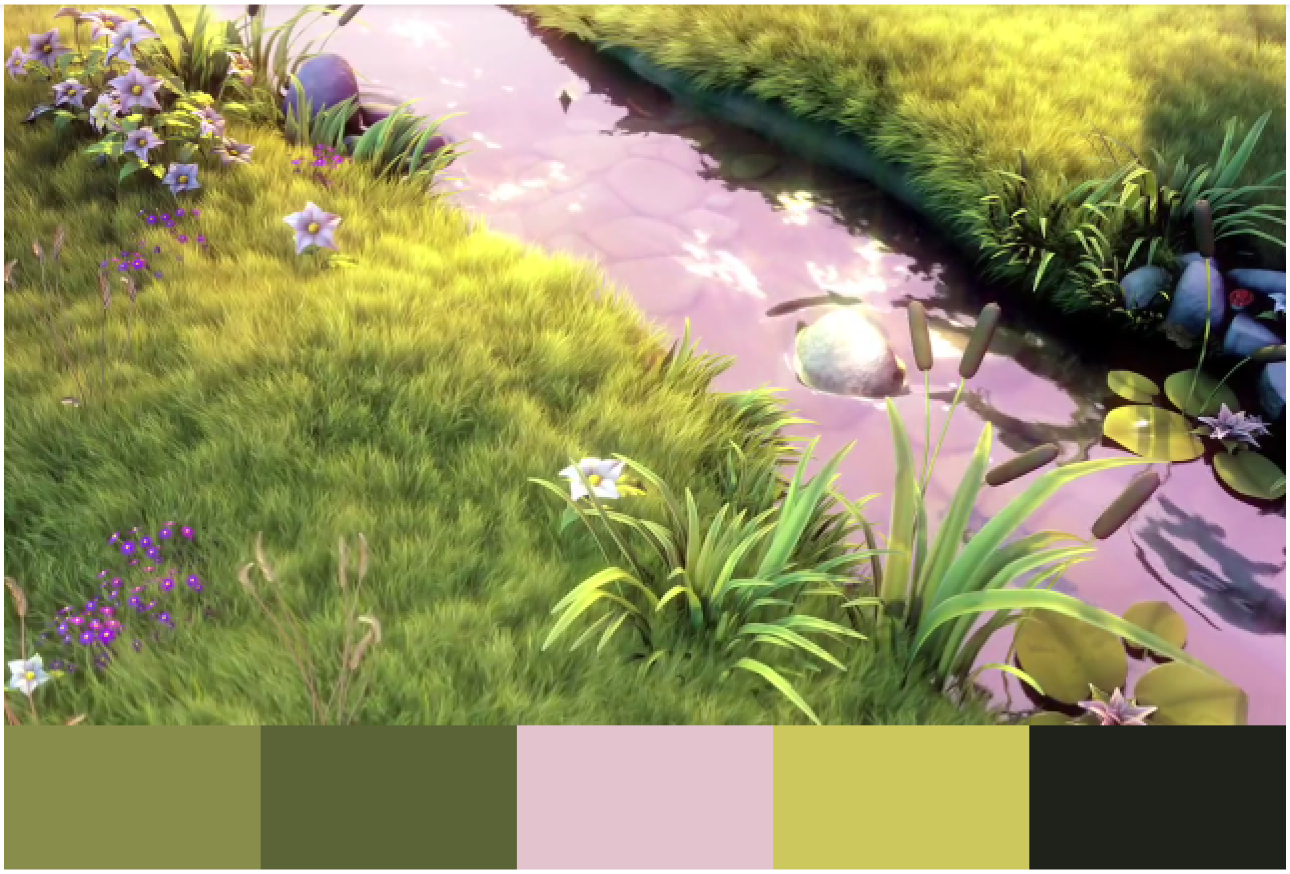
2

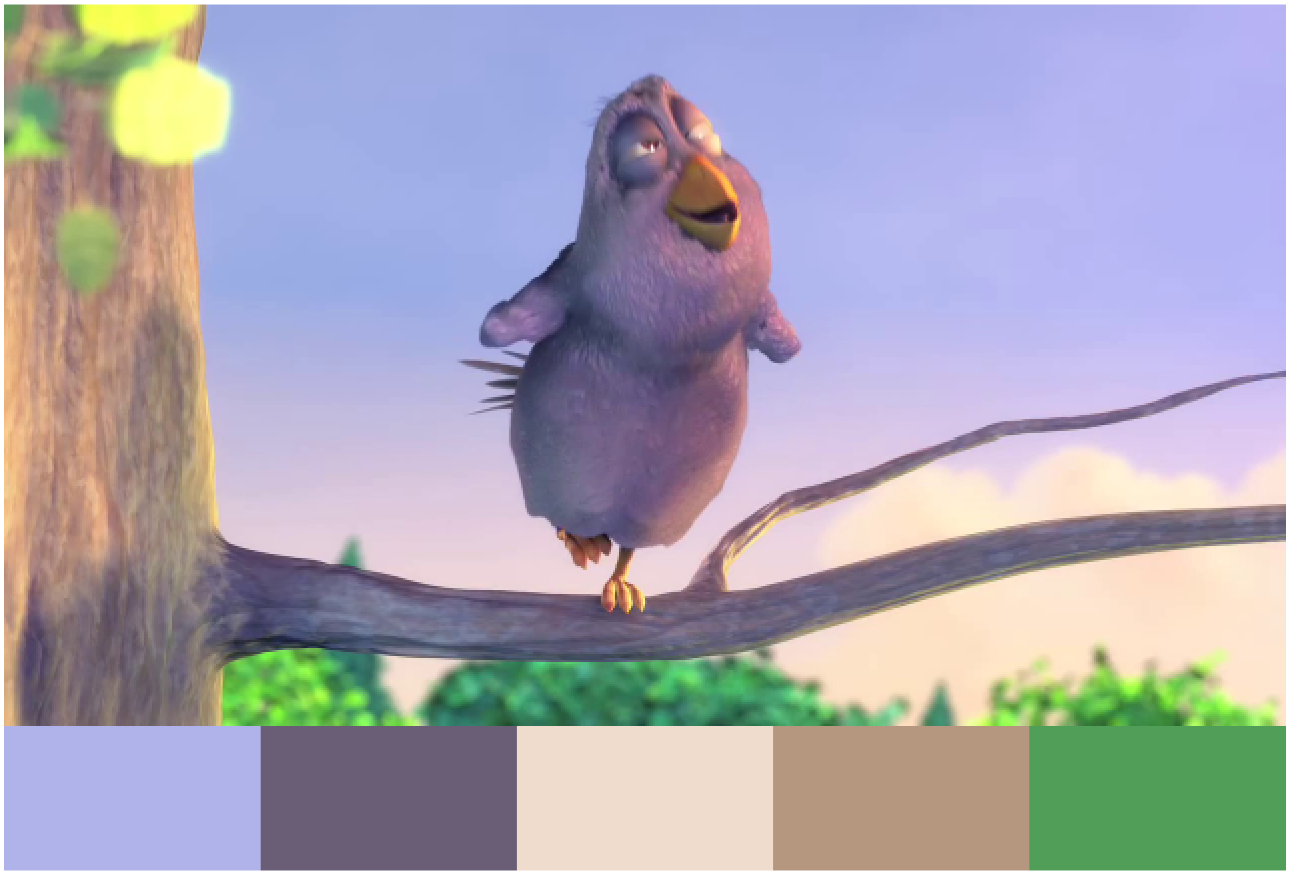
1. Convert an image to a list of pixels `.reshape(-1, 3)`
2. Cluster pixels using kmeans
3. Sort the clusters from largest to smallest
4. Use the cluster center as a color



Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up





Code:

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up

```
cap = cv2.VideoCapture("BigBuckBunny.mp4")
n_clusters = 5

while True:
    status, image = cap.read()
    if not status:
        break

    # to reduce complexity resize the image
    data = cv2.resize(image, (100, 100)).reshape(-1, 3)
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
    flags = cv2.KMEANS_RANDOM_CENTERS
    compactness, labels, centers = cv2.kmeans(data.astype(np.float32),
n_clusters, None, criteria, 10, flags)

    cluster_sizes = np.bincount(labels.flatten())

    palette = []
    for cluster_idx in np.argsort(-cluster_sizes):
        palette.append(np.full((image.shape[0], image.shape[1], 3),
fill_value=centers[cluster_idx].astype(int), dtype=np.uint8))
    palette = np.hstack(palette)

    sf = image.shape[1] / palette.shape[1]
    out = np.vstack([image, cv2.resize(palette, (0, 0), fx=sf, fy=sf)])

    cv2.imshow("dominant_colors", out)
    cv2.waitKey(1)
```

Also you may consider using other distances and color spaces. For example the L2 distance with the LAB color space is better reflects how the person perceives the color similarity.

https://en.wikipedia.org/wiki/CIELAB_color_space#Perceptual_differences

Images taken from the video "Big Buck Bunny": https://peach.blender.org/

Share Follow

answered Sep 22 at 22:13
D u1234x1234
1,646 1 1 7

you are a GOAT, i was using kmeans at the begginig, but dint think about reduce the images so kmeans can work a bit faster – plus Sep 23 at 4:55

but i dont want to use all you code lol, just want until centers, where i get it from kmeans, then using a for loop i print the colors, the problem is that doing it in that way its too fast and its not consistent as yours, only thing is that i dont understand the code after kmeans function and i dont want to copy and use all the code like i wrote it haha – plus Sep 23 at 5:07

You can skip the last part which is used just for visualization. To print colors sorted by dominance use something like this: `for cluster_idx in np.argsort(-cluster_sizes):`

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up

fast, changes the colors really quick – plus Sep 23 at 6:20

Yes you will get an array with dominant colors. The colors are changing for two reasons: 1. Frames are changing 2. Kmeans is a randomized algorithm. You can try more "fixed" approach - 1. use a predefined set of possible colors 2. apply 1-nearest neighbor (query=all pixels; index=predefined colors) and count the number of pixels assigned to the specific predefined color and sort by counts. You can check on how to assign a pixel to the list of colors here stackoverflow.com/questions/73666119/... – u1234x1234 Sep 23 at 16:48

Join Stack Overflow to find the best answer to your technical question, help others answer theirs.

Sign up