A World Travel

By Chromakey

20211060 오서현

20201043 김선경

20190365 김찬주

Github : https://github.com/hyunneee/vmp_travel

Algorithm

1. Import all necessary libraries **(numpy, opencv etc)**
2. Load the target image(or video) and background. **(cv2.VideoCapture(), cv2.imread())**
3. Resize the images and the videos to the same size **(cv2.resize())**
4. Load the upper and lower BGR values of the green color. **(np.array([B,G,R]))**
5. Change the color from BGR to HSV. **(cv2.cvtColor(image or frame, cv2.COLOR_BGR2HSV))**
6. Apply the mask and then use bitwise_and Subtract bitwise_and from the original green screen image. **(cv2.inRange(), cv2.bitwise_and())**
7. Check for matrix value 0 after subtraction and replace it by the second image. **(np.where())**
8. You get the desired results. **(cv2.imshow())**

pseudo code for the algorithm

```
import cv2
import numpy

filename = path
lvideo = cv2.VideoCapture(filename/video_file)
back = cv2.VideoCapture(filename/backgrund_file)

while loop:
        load_video, frame = video.read()
        load_back, b_frame = back.read()

        resize frame, b_frame

        hsv = cv2.cvtColor()

        lower_green =
        upper_green =

        mask : hsv value between lower_green and upper_green

        res : mask value in frame using cv2.bitwise_and
        f = frame - res
        if f == 0:
                f = b_frame

        cv2.imshow(frame)

        if cv2.waitKey() == 27: # using ESC
                break
video.release()
```

Description of the algorithm

This is an algorithm for changing the background in a video (or image) through chroma-keying. To create a mask video, we will crop the background area of the video (taken on the green or blue screen) and change the background in this area.

The algorithm uses the numpy and opencv libraries. The original image (*chroma-keying.mp4*) is the video with a constant green background. In order to apply another video (*road.mp4*) as the background, the frame sizes of them must be the same. Therefore, the frame size is unified through cv2.resize(). (1920x1080)

Next, we define a mask by detecting the background area. For this, we convert from BGR to hsv color space. The 'hsv' is a method to specify colors with hue, saturation, and value, and it is convenient to obtain a specific color area.

Then, cv2.inRange() is used to specify the green background area. Input the image as the first factor of this function and set the minimum and maximum values for the range with the second and third factors. It returns a video (*mask*) to assign set range to 255 (white) and non-set range to 0 (black).

Using the mask, we apply a background image. Here we use cv2.bitwise_and() for bit operations. Since the mask is a binary video consisting of 0 and 255, bit operation is possible. The 'and' operation returns 1 if both bits are 1. So, it can be seen that *res* represents the green screen area. After subtracting this part (*res*) from the original image (*frame*), use np.where() to check for matrix value 0 after subtraction and replace it by the background image (*b_frame*).

The reason we have chosen the algorithm.

We researched two ways to apply a background image to a video using mask operations, cv2.copyTo() and cv2.bitwise_and(). We thought the latter of the two shows more clear images.

Code

```python

import cv2 as cv
import numpy as np

filename = '/Users/seohyunoh/Documents/python/VMP/chroma-key/chroma-keying.mp4'
cap = cv.VideoCapture(str(filename))
# image = cv.imread("eiffel-tower.jpg")
backv = cv.VideoCapture('/Users/seohyunoh/Documents/python/VMP/chroma-key/알고리즘 아이디어/road.mp4')
while(1):
    # Take each frame
    _, frame = cap.read()
    back, b_frame = backv.read()
    frame = cv.resize(frame, (1920, 1080))
    b_frame = cv.resize(b_frame, (1920, 1080))
    # Convert BGR to HSV
    hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)
    # define range of blue color in HSV
    lower_green = np.array([40, 40, 0])
    upper_green = np.array([70, 255, 255])
    # Threshold the HSV image to get only blue colors
    mask = cv.inRange(hsv, lower_green, upper_green)

    # Bitwise-AND mask and original image
    res = cv.bitwise_and(frame,frame, mask= mask)
    f = frame - res
    f = np.where(f == 0, b_frame, f)

    cv.imshow('frame',frame)
    cv.imshow('mask',f)
    # cv.imshow('res',res)
    k = cv.waitKey(5) & 0xFF
    if k == 27:
        break
cap.release()
cv.destroyAllWindows()
```

Sources

- ✓ https://www.geeksforgeeks.org/replace-green-screen-using-opencv-python/
- ✓ https://deep-learning-study.tistory.com/134
- ✓ https://medium.com/fnplus/blue-or-green-screen-effect-with-open-cv-chroma-keying-94d4a6ab2743