[VMP] Team Project

DIGITAL MATTING

A World Travel

By Chromakey

20211060 오서현

20201043 김선경

20190365 김찬주

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

Github Web Page : https://hyunneee.github.io/vmp_travel/

Project Purpose

The purpose of this project is to implement chroma keys using opencv.

pseudo code for the Initial 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 Initial 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.

References

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

Experiment

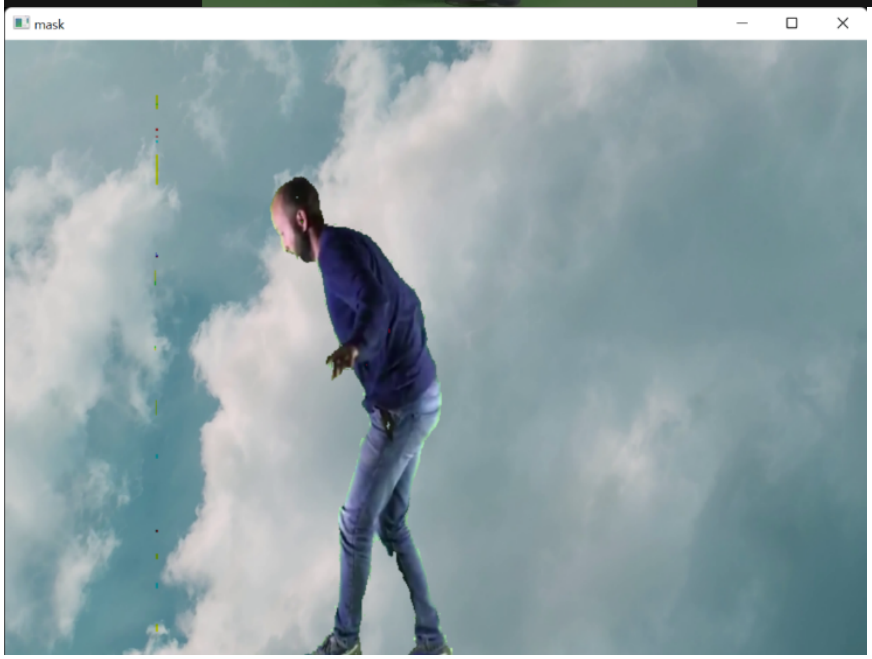
To verify that our code runs well, we tried to experiment with the code through various input images. Images taken from a green background that can be found through research can be divided into three categories.

1. Source taken in real life
2. Source whose existing background was removed and green background was applied.
3. Various effects sources with a green background Image taken in where the background scene is of a unique color

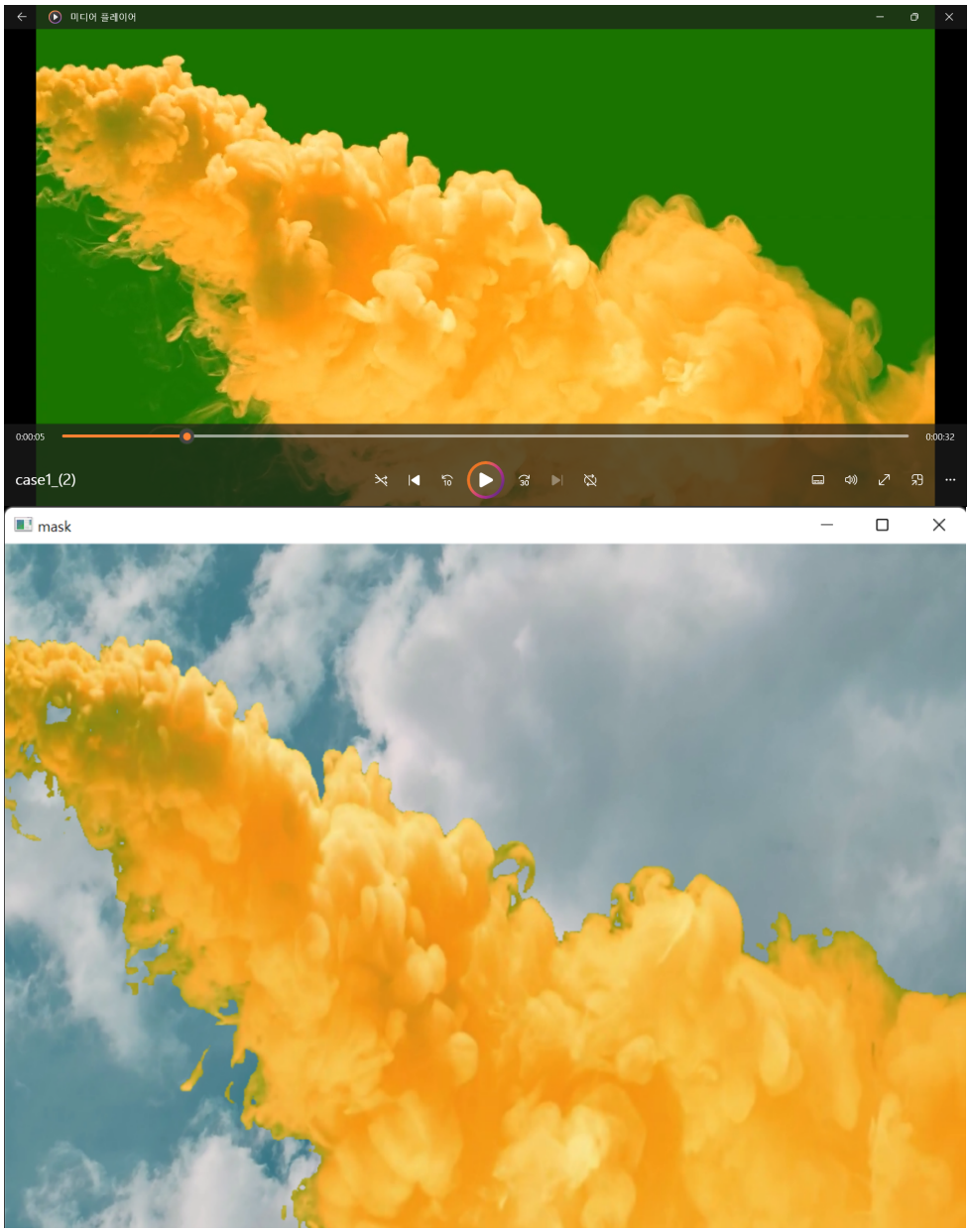
Therefore, we tried to experiment with our code by selecting two images for each category.

1. Source taken in real life

- Case1_(1)



- Case1_(2)



It can be confirmed that both images are detected clearly without additional code modification.

2, Source whose existing background was removed and green/blue background was applied.

- Case2_(1) Apply Green

Result

- Case2_(2) Apply Blue

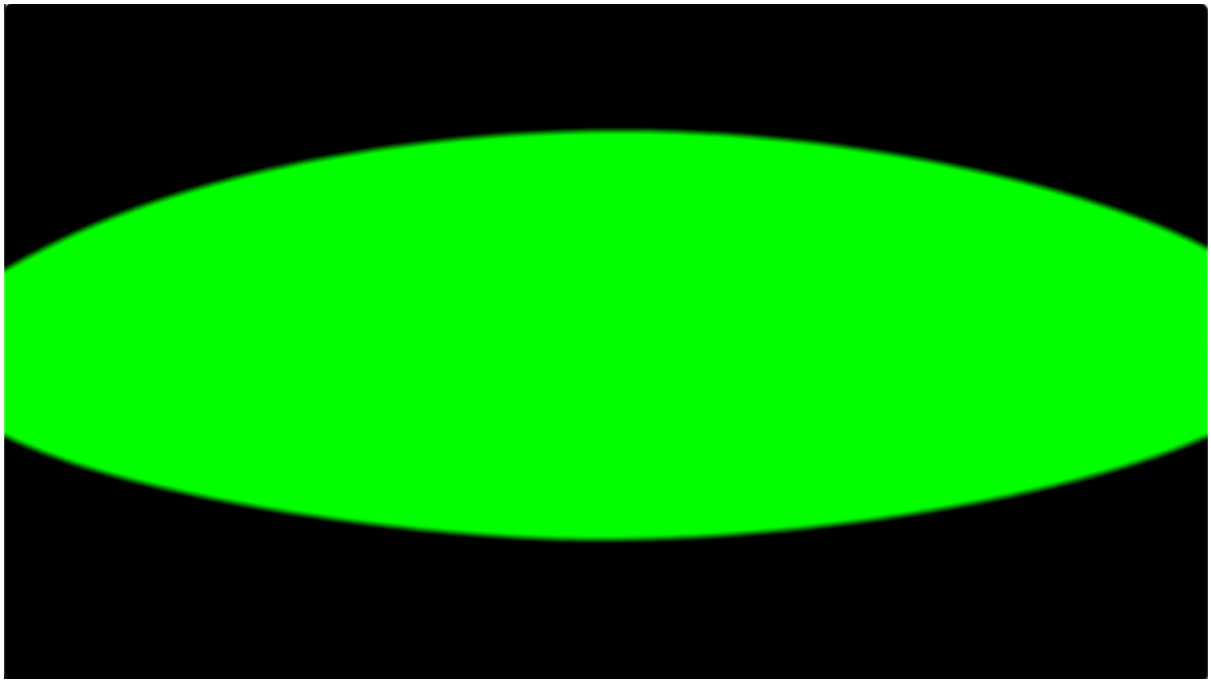
```
# define range of blue color in HSV
lower_blue = np.array([110, 50, 50]) # h
upper_blue = np.array([130, 255, 255])
```

Change the blue range of HSV because blue, not green, must be removed.

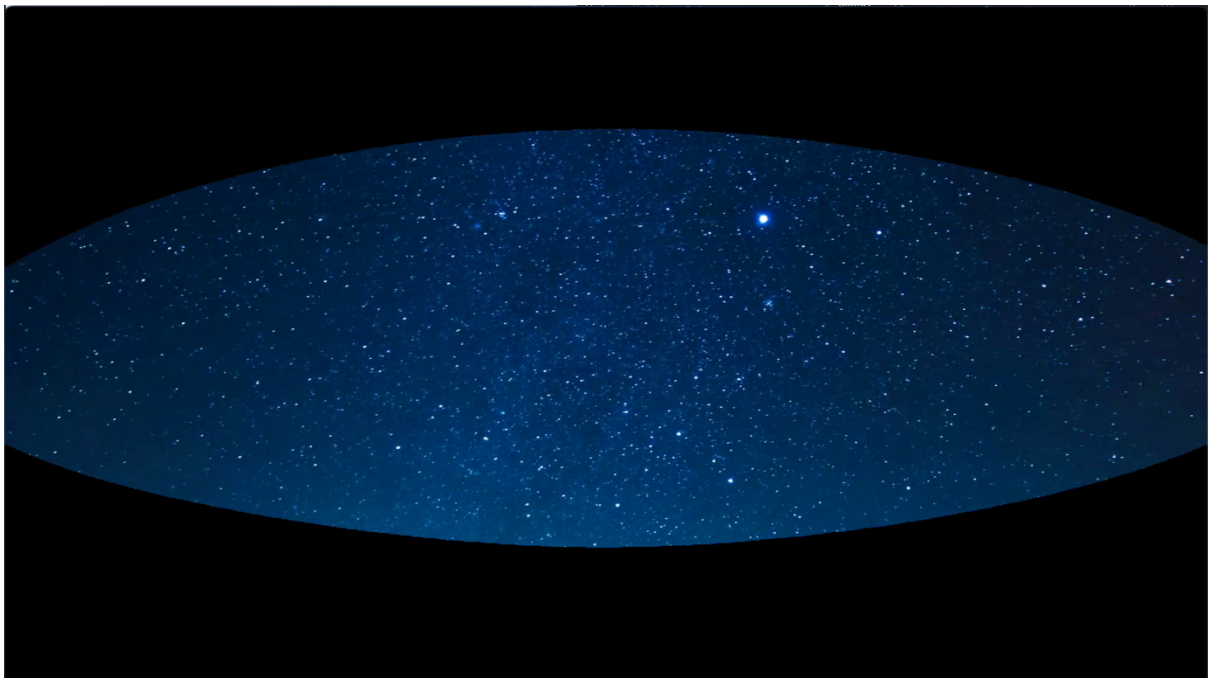
Result

3. Various effects sources with a green background Image taken in where the background scene is of a unique color

- Case3_(1) Blink

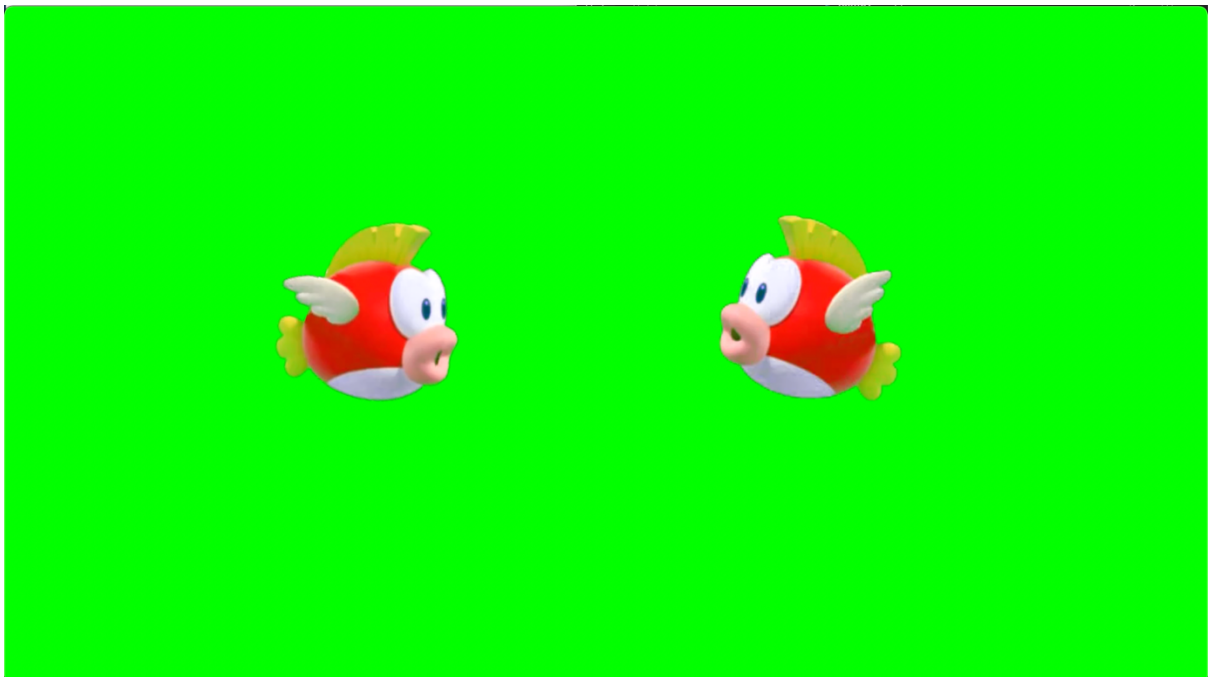


result



It can be confirmed that both images are detected clearly without additional code modification.

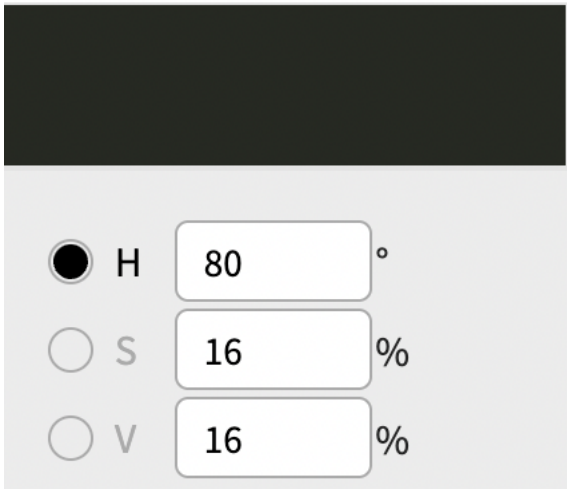
- Case3_(2) Fishes



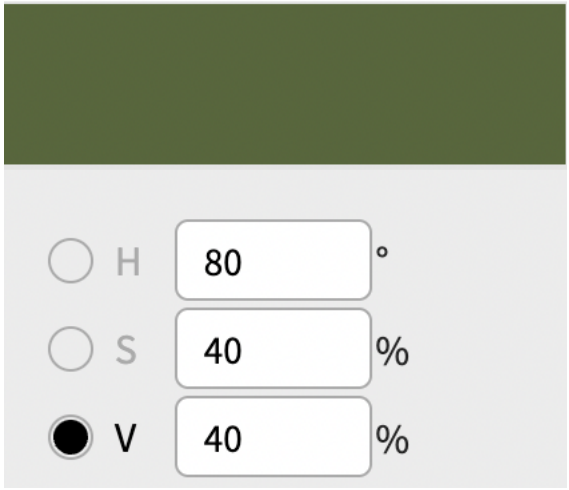
First Trial(lower_green = [40,40,40]

When applied to this video, there was a part where chroma key was not applied. So We searched the green range in HSV once again. When the range of lower green was changed from [40, 40, 40] to [50,100, 100], it was found that it was well applied.

Second Trial(lower_green = [50,100,100])



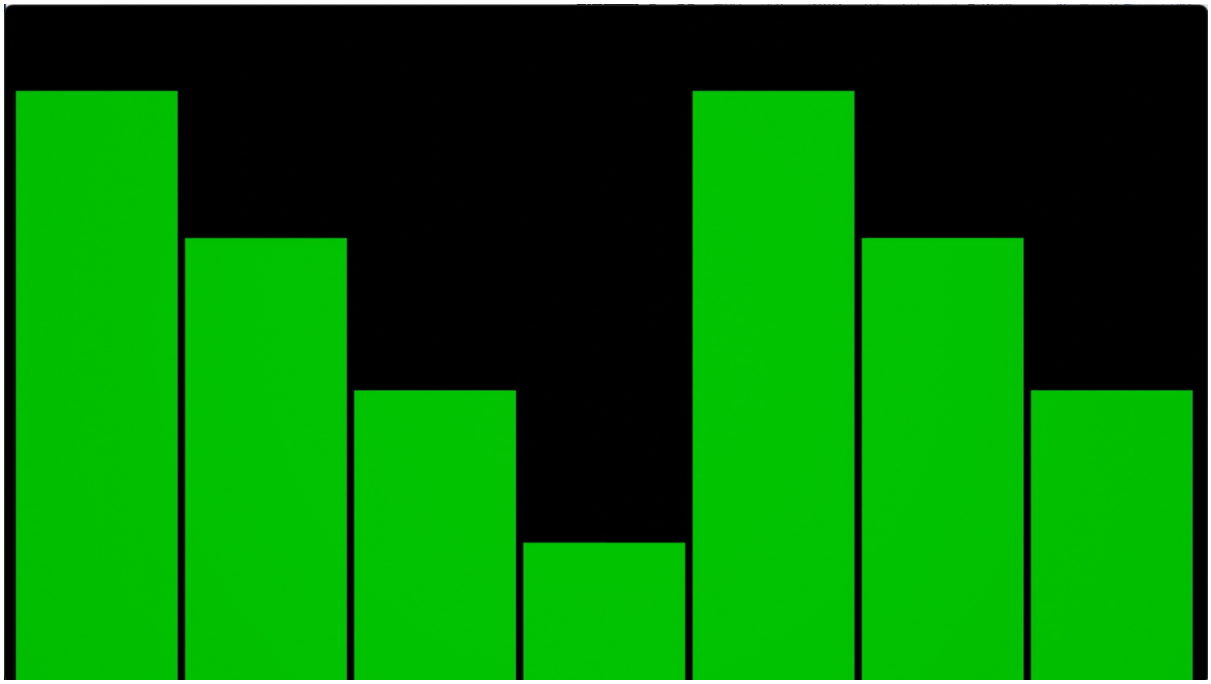
in [40, 40, 40], the lower_green color is like this.



in [50,100,100], the lower_green color is like this.

In opencv, color range is different from the range of the picture because the range of hue is 180 degrees and the range of saturation and value is 256.

- Case3_(3) Slider



result

- Case3_(4) Snow effect

result

Problems in the process of project & Solution

- Initially, we used cv2.copyTo() in the process of chroma-keying through mask operations. The image and background were combined, but the boundary between the two images that were merged was not seen smooth. We could solve this problem by adjusting the value of lower_green and upper_green, but it was still not satisfactory. Therefore, after further investigation, we found and used the cv2.bitwise_and() function, which more naturally merges the image, so we solved the problem using this function.

- The code is played based on the background image, and the video ends when the background video ends not target video. The reason why this phenomenon occurred is that the lengths of the background video and the green screen video were different. So to solve this problem, we decided to put on a loop that plays again even after the background video is over.

  At this time, We use cv2.CAP_PROP_POS_FRAMES, cv2.CAP_PROP_FRAME_COUNT. The former is a variable that knows the current number of frames of a video, and the latter is a variable that counts the number of frames of the video. If these two numbers are the same, it means that the video is over, and at this time, the current number of frames was corrected to zero to perform an video loop.

- there was a problem that the image was played slowly. The reason is that the number of frames in the image was different. Looking at the code, I found that the frames of the recorded image were not matched according to the number of frames of the image, and were fixed at 30 frames. Therefore, we modified this code. At this time, We use cv2.CAP_PROP_FPS that represent FPS of video.

- When conducting the experiment, the experiment was conducted by continuously changing the path of the videos, and We wanted to allow it to be determined at the terminal. For this, we used argv. argv is an array that stores characters entered based on the blanks in the terminal. In order to use this, sys was imported, and green screen video, background video, and result video were stored in order and set as variables.

Core Algorithm for Chromakey

1. Import all necessary libraries (numpy, opencv etc)
2. The code is executed along with the green screen image, the background image, and the output image at the terminal.
3. It stores videos using cv2.VideoCapture, stores fps, and sets variables to record video using cv2.VideoWriter.

In while loop

4. Load the target image(or video) and background. (cv2.VideoCapture(), cv2.imread())
5. Resize the images and the videos to the same size (cv2.resize())
6. Load the upper and lower BGR values of the green color. (np.array([B,G,R]))
7. Change the color from BGR to HSV. (cv2.cvtColor(image or frame, cv2.COLOR_BGR2HSV))
8. Apply the mask and then use bitwise_and Subtract bitwise_and from the original green screen image. (cv2.inRange(), cv2.bitwise_and())
9. Check for matrix value 0 after subtraction and replace it by the second image. (np.where())
10. if target video(green screen video) were done, break.

11 if background video were done, restart (cv2.CAP_PROP_POS_FRAMES, cv2.CAP_PROP_FRAME_COUNT)

11.release every videos. (.release(), cv2.destroyAllWindows())

12.You get the desired results. (cv2.imshow())

Final Code

```python
import cv2 as cv # opencv
import numpy as np # numpy
import sys # for terminal command

# cammand : $ python chroma_key.py(argv[0])  input_video.mp4(argv[1])  background.png(argv[2]) output_video.mp4(argv[3])

args = sys.argv[1:] # array in range(1,)
target_v_dir = sys.argv[1] # green screen (target) video
back_v_dir = sys.argv[2] # background video
output_v = sys.argv[3] # output video name

width , height = 1920, 1080 # set window size
video = cv.VideoCapture(target_v_dir)
backv = cv.VideoCapture(back_v_dir)
fps = video.get(cv.CAP_PROP_FPS)
recorder = cv.VideoWriter(output_v,
                          cv.VideoWriter_fourcc(*'MP4V'),
                          fps,
                          (width,height))

while(1):
    # Take each frame
    load_video, frames = video.read()
    load_back, b_frames = backv.read()

    # resize the both videos
    frames = cv.resize(frames, (1920, 1080))
    b_frames = cv.resize(b_frames, (1920, 1080))

    # Convert BGR to HSV
    hsv = cv.cvtColor(frames, cv.COLOR_BGR2HSV)

    # define range of green color in HSV
    lower_green = np.array([55, 50, 70])
    upper_green = np.array([89, 255, 255])
    # [89, 255, 255], [55, 50, 70] which is the proper green color(hsv) range in target video
    mask = cv.inRange(hsv, lower_green, upper_green) # set mask

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


    cv.imshow('Chromakey_result',f) # show the result in window
    recorder.write(f) # record result video

    if not load_video: #if video done, break
        break

    if backv.get(cv.CAP_PROP_POS_FRAMES) == backv.get(cv.CAP_PROP_FRAME_COUNT): # if back done, restart
        backv.set(cv.CAP_PROP_POS_FRAMES,0)

    if cv.waitKey(1) == 27: # if pree esc, break
        break

video.release()
backv.release()
recorder.release()
cv.destroyAllWindows()
```

Final Result

Green screen video

Result