TEAM
STRAWBERRY

3 AWESOME THINGS YOU CAN DO WITH CHROMA-KEY

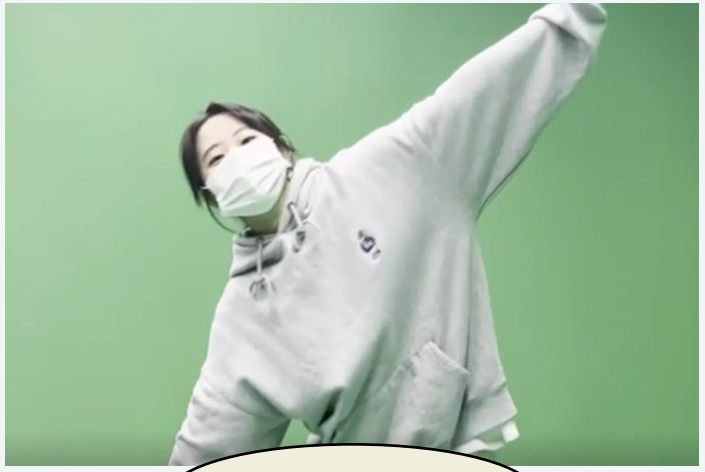
MAS2011 • Introduction to Visual Media Programming
Interim Report

20192105 윤 찬
20221054 심 유 림
20221110 정 수 겸

https://github.com/VMPstrawberry/VMP_TeamProject_Chroma-Key

3 AWESOME THINGS YOU CAN DO WITH CHROMA-KEY

1. Swimming

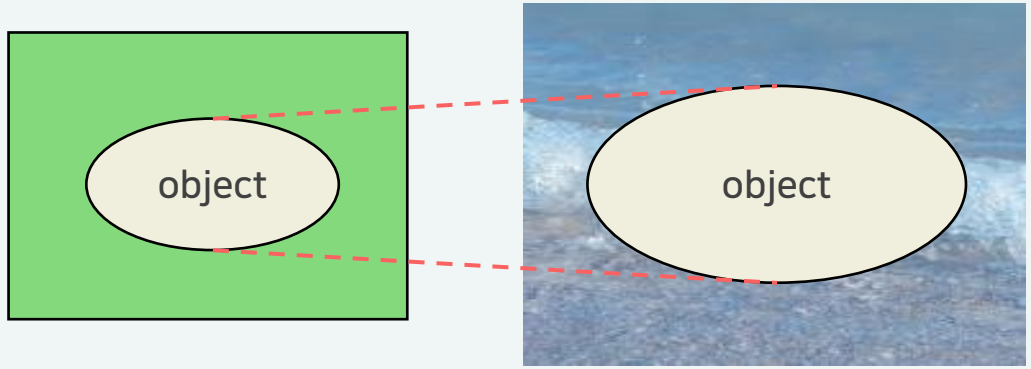


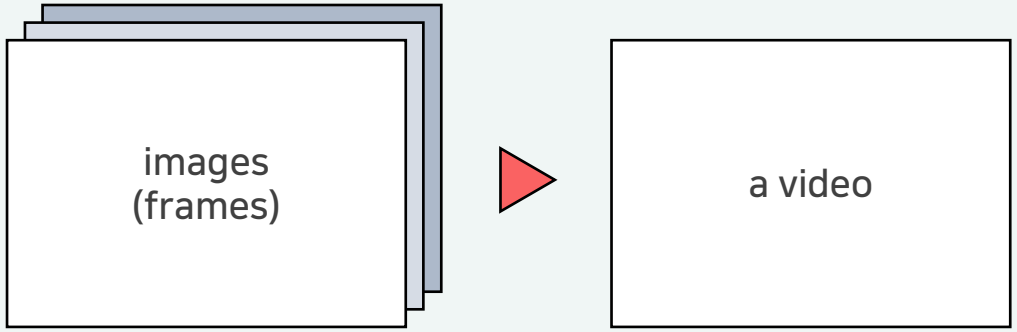
2. Flying

3. Working out

INDEX

001-1 Algorithm Summary





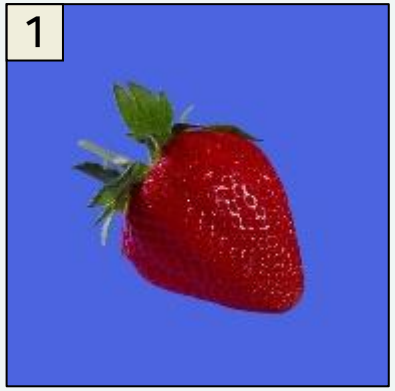
Our goal is to insert a suitable background for each chroma key video we made. We will go through the process of extracting the object from the green background.

Basically, a video consists of several frames. Therefore, we first thought about how to apply the chroma key on a single frame by using an image. Based on this, we developed the algorithm step by step to find a way to apply it to a video with multiple frames.

001-1 Algorithm Summary

GOAL To extract the target from the chroma key video and put it on a new background image.

1

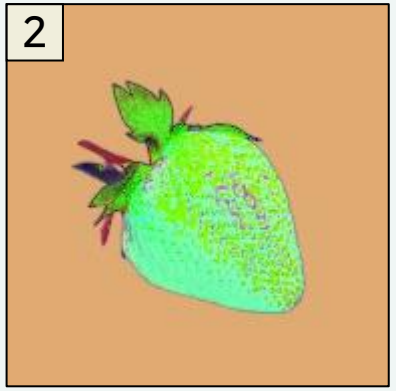


<src>

Prepare a chroma key video of which you want to change the background and get each frame from the video.

>>

2

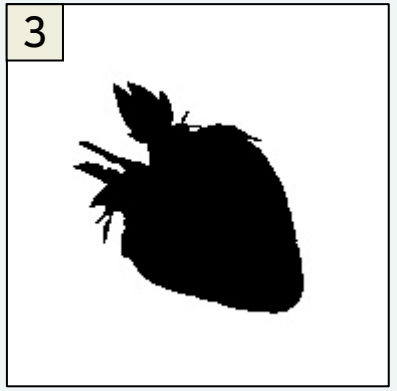


<hsv>

Convert the BGR image to HSV image. Set the color range of the background color you want to extract.

>>

3



<mask>

Change the whole pixel value of the selected part to 255(white). For the other part, change it to 0(black).

>>

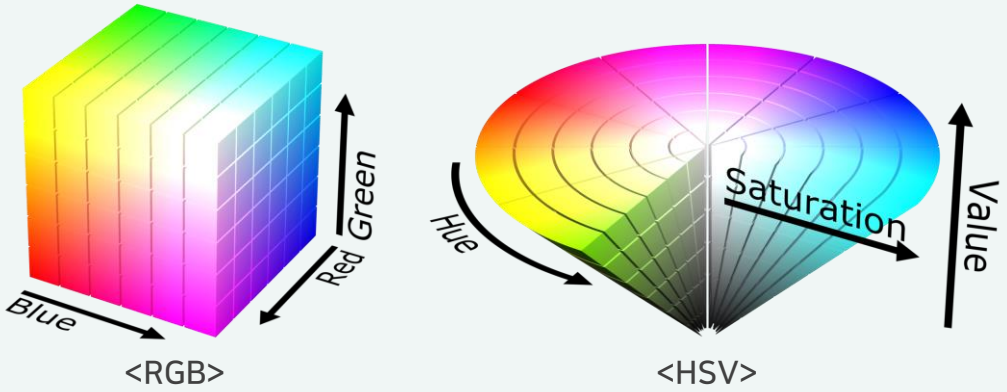
4

<dst>

Extract corresponding part from <src> which has the same index as the selected part of <mask> and add it on the new background.

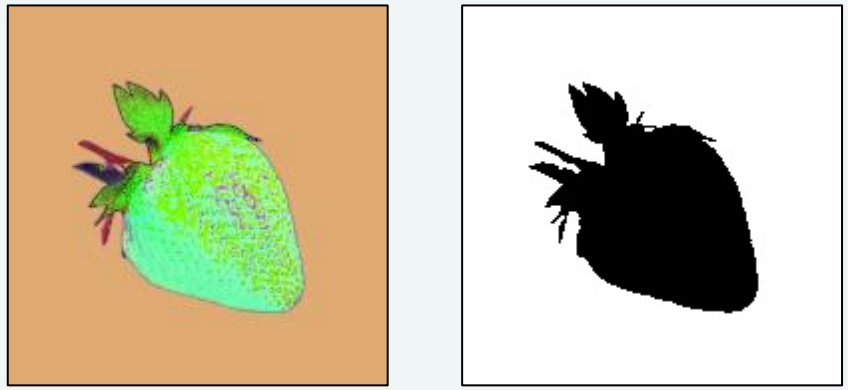
001-2 Main Ideas

① Converting BGR → HSV



- HSV is more intuitive than RGB or BGR since we can adjust hue, saturation, and brightness values separately.
- It is much easier to guess what color it will be when we change a specific value.

② Masking



- By simplifying the values of pixels, we can conveniently use Boolean Indexing.
- Since the value of each pixel is either 255(white) or 0(black), it is easy to invert the selected area.

002-1 Algorithm Development

1. cv2.cvtColor(src, cv2.COLOR_BGR2HSV)

→ Converts the color of the image to HSV values.

2. cv2.inRange(hsv, hsv_lower, hsv_upper)

hsv_lower : minimum value of the color range
hsv_upper : maximum value of the color range

mask = cv2.inRange(hsv, hsv_lower, hsv_upper)

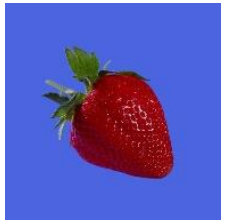
→ Returns the image having a black background and white object.
→ The image would only have the pixel value of 255 or 0.

3. Masking

pixel_value = mask
dst[pixel_value == 0] = src[pixel_value == 0]

→ Simple indexing

```python
import numpy as np
import cv2

# Image uploading
im = cv2.imread('Images/strawberry.jpg', cv2.IMREAD_COLOR)
src = cv2.resize(im, (190,190))
hsv = cv2.cvtColor(src, cv2.COLOR_BGR2HSV)

# Background image uploading
bg = cv2.imread('Images/background.jpg', cv2.IMREAD_COLOR)
dst = bg[0:190,0:190]

# print(src.shape) --> (190, 190, 3)
# print(dst.shape) --> (190, 190, 3)
# print(hsv[50,50]) --> [115, 170, 224]

# hsv_lower = np.array([80, 50, 100])
# hsv_upper = np.array([150, 255, 240])

hsv_lower = np.array([115, 170, 224])
hsv_upper = np.array([115, 170, 224])

mask = cv2.inRange(hsv, hsv_lower, hsv_upper)

pixel_value = mask # just renaming

dst[pixel_value == 0] = src[pixel_value == 0]
```



https://pin.it/5cJJwa8

002-1 Algorithm Development

1 The size of the source(a frame of the video) should be the same with of the background image.

(X)

<source image>

<background image>

(O)

<source image>

<background image>

2 For the cleaner result, the color range should be as wide as possible.

<bad example>

<good example>

```
hsv_lower = np.array([115, 170, 224])
hsv_upper = np.array([115, 170, 224])
```

(X)

```
hsv_lower = np.array([80, 50, 100])
hsv_upper = np.array([150, 255, 240])
```

(O)

002-2 Attempt 1: How to remove Afterimages

>> Problem

The previous frame does not disappear and remains. We intended the result of a video that moves without afterimages. Why does the previous image remain and how can I solve this problem?

002-2 Attempt 1: How to remove Afterimages

```python
src = cv2.VideoCapture('Videos/flying.mp4')
im = cv2.imread('Images/sky.jpg', cv2.IMREAD_COLOR) # Photo by
dst = cv2.resize(im, (900,720))  ①

src_fps = src.get(cv2.CAP_PROP_FPS) # 29.995496171745984
src_count = src.get(cv2.CAP_PROP_FRAME_COUNT) # 333.0
src_width = src.get(cv2.CAP_PROP_FRAME_WIDTH) # 900.0
src_height = src.get(cv2.CAP_PROP_FRAME_HEIGHT) # 720.0

hsv_lower = np.array([50, 40, 108])
hsv_upper = np.array([70, 130, 223])

recorder = cv2.VideoWriter("Videos/Flying in the sky.mp4",
                           cv2.VideoWriter_fourcc(*'MP4V'),
                           src_fps, (900, 720))
while src.isOpened():
    ret, frame = src.read()

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, hsv_lower, hsv_upper)
    mask_inv = cv2.bitwise_not(mask)
    pixel_value = mask_inv
    dst[pixel_value > 0] = frame[pixel_value > 0]  ②

    cv2.imshow('src', frame)
    cv2.imshow('dst', dst)

    recorder.write(dst)  ③
```

Cause

We just uploaded the background image once at ①.

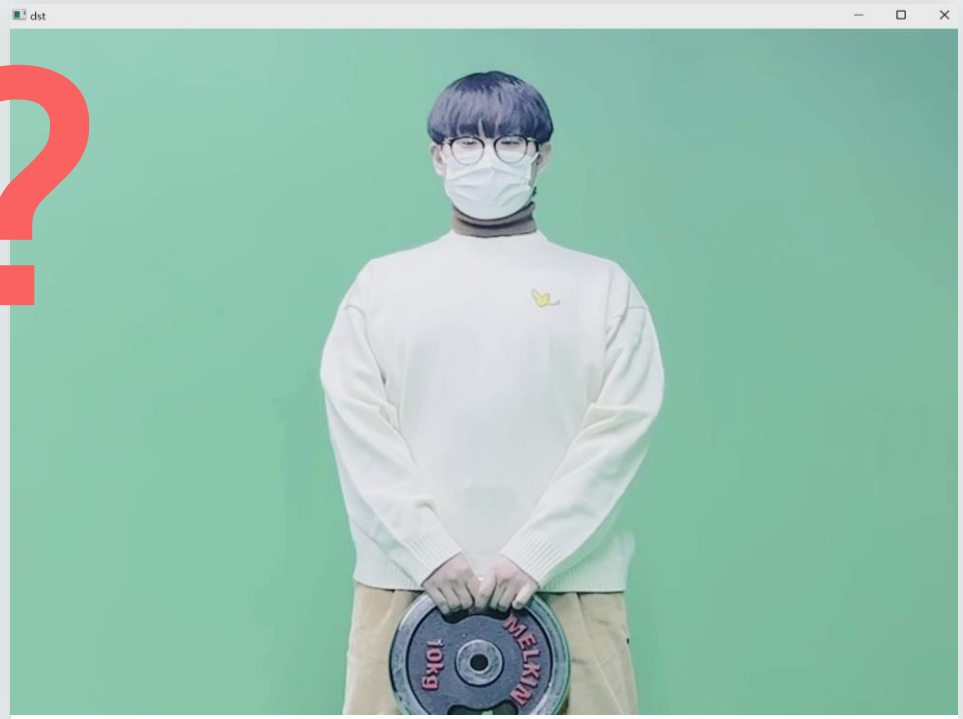
However, as we changed some pixels at ② and recorded it at ③, the modified pixels were continuously overlapping.

Solution

By replacing ① into the *while loop*, we could keep updating the background image every time the frame changes. Now, the afterimages does not appear.

```python
while src.isOpened():
    ret, frame = src.read()

    dst = cv2.resize(im, (900,720))
```

002-3 Attempt 2 : How to set a proper Color Range



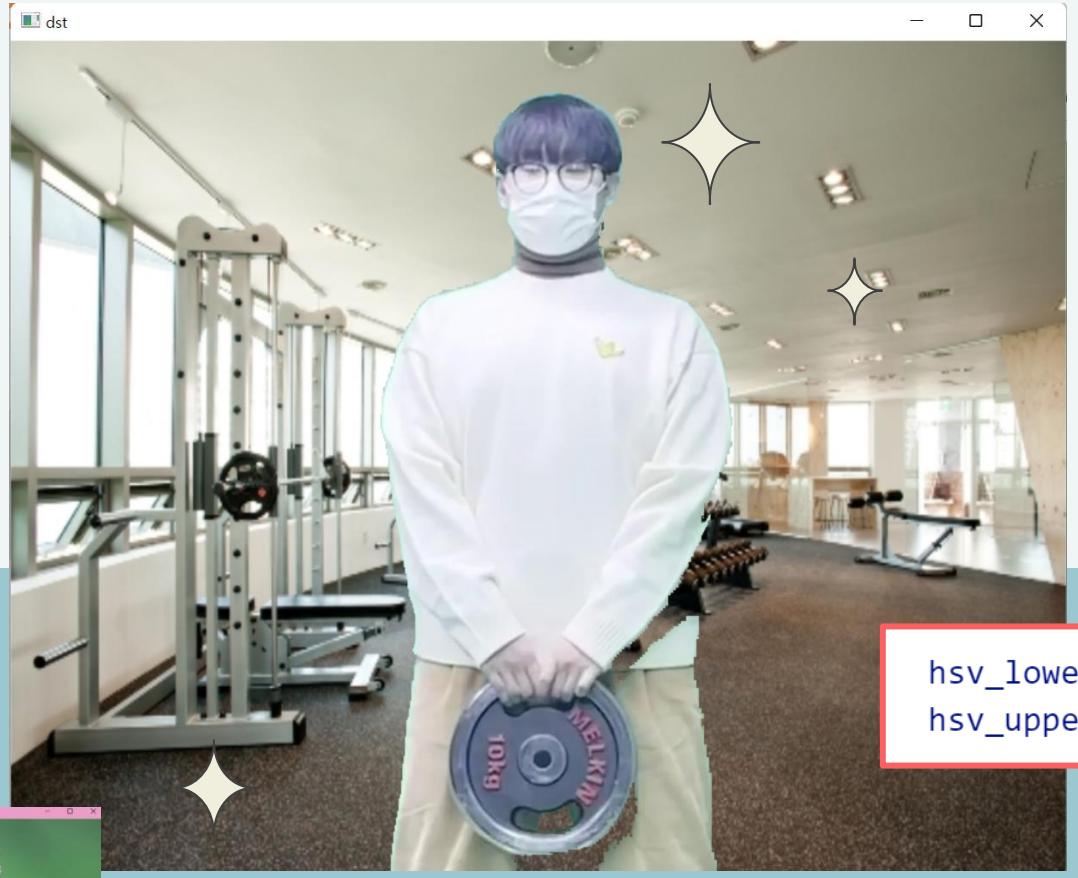
≫ Problem

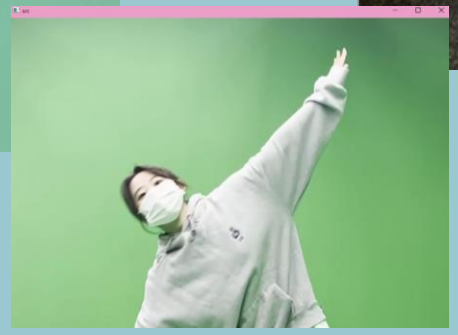
Actually, this is the result of applying the revised code to another video. There are no afterimages, however, the chroma key does not seem to be applied to this video at all. What is the problem?

002-3 Attempt 2 : How to set a proper Color Range

Cause

Only the subject should be extracted by selecting the green part, But the color range did not fit in this video, causing the chroma key not to be applied properly.





As you can see here, the background colors of the two videos are slightly different.

Solution

We repeated the process of checking the result by raising the H upper value by 5. The result on the center is the result of setting the upper limit value to 85 which is the best result among the process of raising by 5 from 70 to 100.

```
hsv_lower = np.array([50, 40, 108])
hsv_upper = np.array([90, 130, 223])
```

```
hsv_lower = np.array([50, 40, 108])
hsv_upper = np.array([85, 130, 223])
```

```
hsv_lower = np.array([50, 40, 108])
hsv_upper = np.array([80, 130, 223])
```

```
hsv_lower = np.array([50, 40, 108])
hsv_upper = np.array([75, 130, 223])
```

...

```python
import numpy as np
import cv2

# Upload source video and background image
src = cv2.VideoCapture("Videos/flying.mp4")
background_image = cv2.imread("Images/sky.jpg", cv2.IMREAD_COLOR)
# Photo reference: Photo by Arteum.ro on Unsplash

# Get the information of the src video
src_fps = src.get(cv2.CAP_PROP_FPS) # frame per second
src_width = src.get(cv2.CAP_PROP_FRAME_WIDTH)
src_height = src.get(cv2.CAP_PROP_FRAME_HEIGHT)

# You can check the information here
print(" Information ".center(60, "="))
print(f"fps : {src_fps},    width : {src_width},    height : {src_height}".center(60))
print("="*60)

# Determine the hsv color range
hsv_lower = np.array([50, 35, 133])
hsv_upper = np.array([65, 110, 235])

# Make a recorder to record the result
recorder = cv2.VideoWriter("Videos/result.mp4",
                           cv2.VideoWriter_fourcc(*'mp4v'),
                           src_fps, (int(src_width), int(src_height)))

```

```
while src.isOpened():
    # Get each frame from the video
    ret, frame = src.read()

    if frame is None:
        break
    else:
        # Reset the background
        dst = cv2.resize(background_image, (int(src_width), int(src_height)))

        # Convert BGR to HSV
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        # Convert each frame to black & white image
        mask = cv2.inRange(hsv, hsv_lower, hsv_upper)
        pixel_values = mask # just renaming

        # Cutting and pasting
        dst[pixel_values == 0] = frame[pixel_values == 0]

        # Show before and after
        cv2.imshow('Before (src)', frame)
        cv2.imshow('After (dst)', dst)

        # Record the modified frame
        recorder.write(dst)

        # Repeat every 10 milliseconds
        if cv2.waitKey(10) == 27:
            # Press [Esc] to stop during the process
            # If you do that, the record would not be
            # Please wait for the process itself to st
            break

print("Your video is ready!")
src.release()
cv2.destroyAllWindows()
```




MELKIN
10kg



MELKIN
10kg

003-1 Best result

<Before>

<After>

```
hsv_lower = np.array([40, 30, 108])
hsv_upper = np.array([104, 255, 462])
```

In this part, the swimming subject is extracted and inserted into the deep sea image. By simply adjusting the HSV values, the subject is distinguished quite accurately from the green background.

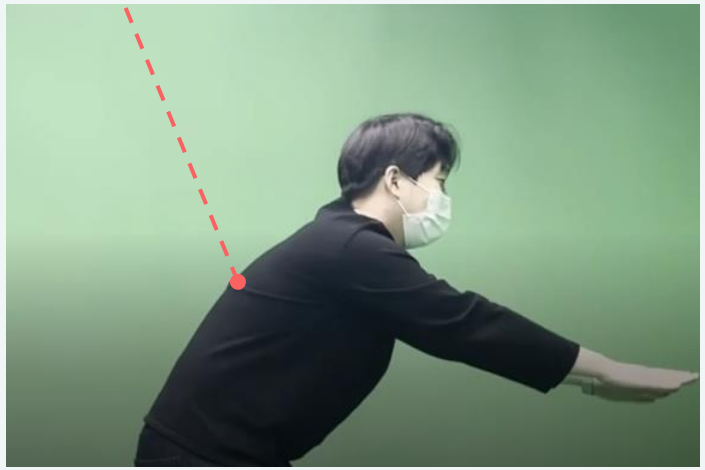
Despite using the same algorithm,
our 3 experiments produced different results. Why?

003-2 Analysis and Limitation

<< As you can see here, chroma key has been applied somewhat incompletely on those two. As the subject moves, some parts still look green or even disappear, which means that it was not completely extracted from the background. On the other hand, in the case as below, chroma key was applied almost perfectly, unlike the previous two cases. We could get a very clear form of subject here. What makes this difference?

003-2 Analysis and Limitation

Dark clothes



Bright clothes

The secret lies in the color of the clothes. The circles above are the colors extracted from the clothes the subjects are wearing in each video. However, as you can see from the color, in the two cases wearing bright clothes, ***green***, which is almost similar to the background color, is extracted from the clothes. Perhaps, the chroma key did not applied properly because the green color of the wall was reflected in the bright clothes during the filming process, and green shadows were formed as the subject moves.

Unfortunately, since we are using a color-picking algorithm, there is no way to do something when the pixel values of those green parts in the background color is the exactly same with those in the subject. Therefore, this is currently our best, and to solve this problem, we should do something to prevent green color from reflecting on our clothes during the filming process.