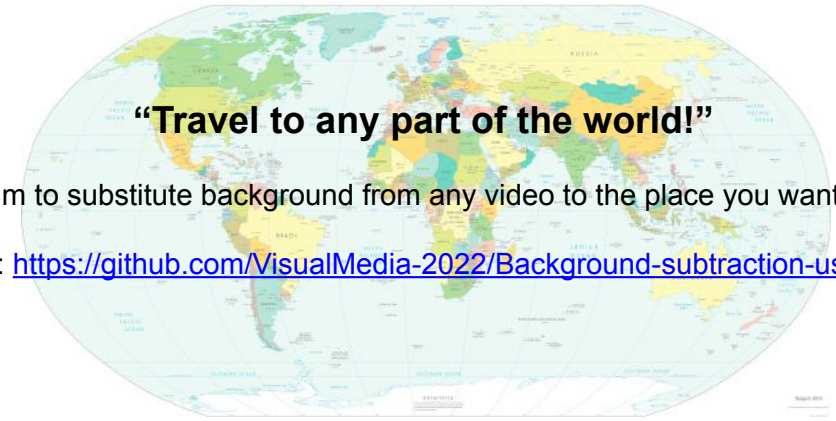


“Travel to any part of the world!”

An algorithm to substitute background from any video to the place you want to travel to

Github link: https://github.com/VisualMedia-2022/Background-subtraction-using-opencv

1. Team Members

Name: Kothari Nandita Manish

Student number: 20192017

Name: Kim Byul

Student number: 20220997

Name: Choi Yunjeong

Student number: 20221129

TP1-2: ALGORITHM SUMMARY

20192017 Kothari Nandita Manish, 20220997 Kim Byul, 20221129 Choi Yunjeong

1. Reason for choosing this algorithm:

First we thought about creating an algorithm that removes background from videos using a blue screen where we would use hsv threshold to find and remove the background. However, since the application of that code would be limited to only blue screen videos, we decided to create an algorithm that could remove and substitute the background of any video where the foreground (moving people/animals/cars) and a stationary background has visible differences in color. We also wanted to make an algorithm which anyone can use. They don't need to know that inputting a background video without the subject is required. That's why we will try to build an algorithm that can remove background from a video even without providing the clean background video.

2. Algorithm Ideas:

A. **Getting foreground and background:** Before putting the new background image, we need to first differentiate the foreground and the background. To do this, we will try two different methods to get the foreground and background as black and white (gray scale) video. After implementing the codes, we will decide which method works the best based on the results.

- **Method 1: will use the method that is explained in this video**: https://www.youtube.com/watch?v=fEi63QB81ek

  > Video requirement: The idea is to have a video which is recorded on a tripod, i.e. the video doesn't move or is shaky.

  > Finding the background pixel values: The algorithm randomly chooses 25-30 (will decide how many frames later while we implement the code to get the best results) and gets the median values of the pixels from all the frames. That median value then becomes the pixel value for the background image that will be used later to create a mask.

  > Converting video to grayscale: we convert the video frames to gray scale before calculating the absolute difference between video frame and background.

  > Determining whether the pixel in the video frame belongs to foreground or background: Now we compare each frame of the video with the background image we calculated above. We calculate the **absolute difference** between the pixels in the frame of the video and the pixels in the background image.Using the opencv threshold function, we **set a threshold** for the absolute difference value between the pixels. If the difference between the pixel's RGB value in a particular position of the frame and the background image is equal to or below the threshold, then the pixel belongs to the background and will return 1, if the difference is greater than threshold, then the pixel belongs to the

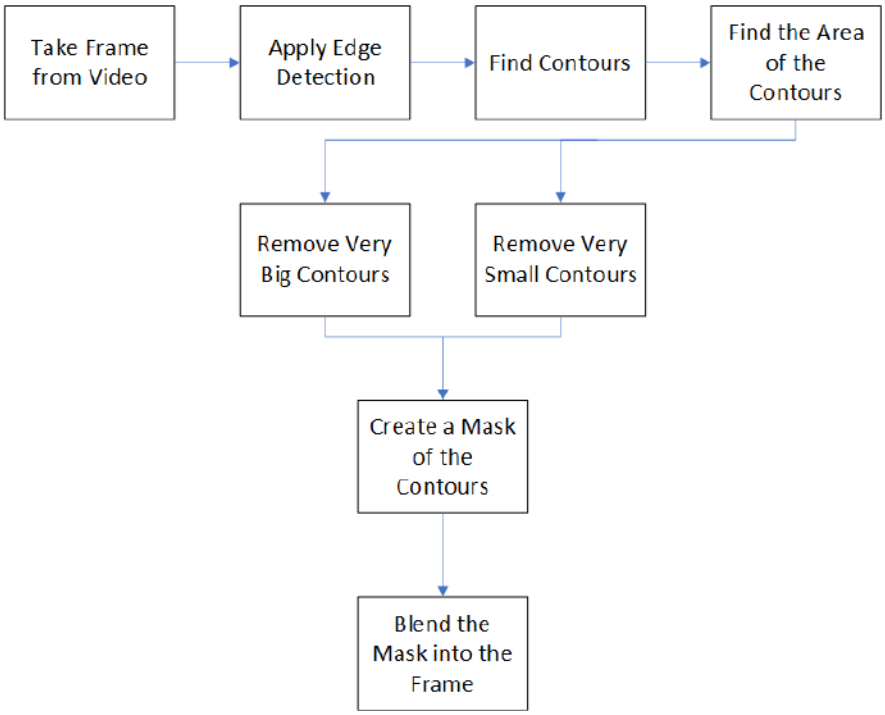
foreground and will return 0. The background is assigned black color and the foreground white color.

- **Method 2: Contouring the object:** inspired by this article: Background Removal with Python. Using OpenCV to Detect the Foreground | by Andrew Udell | Towards Data Science
- https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html

  > Video converting: First we convert the video image to black and white. And to find contour by applying the edge detection function in opencv.

  Contour is a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition(opencv docs, contours).

  > Find Contours area: After finding Contour, we have to calculate the area within the contouring boundary. The foreground or the subject is assigned white color and the background is assigned black color.



B. **Create a mask over the foreground:**

(47) When you don't have a chromakey but want to have an epic background... - YouTube

The idea of this algorithm is like this:

1. grayscale the image (foreground)
2. extract edges of the grayscaled image
3. mask outside part of the edges in black so that the internal part of the edges become white
4. replace white to actual video(foreground video)
5. replace black to background image

> Why we use Black and White (Binary) image?:
In this method, we need to create a binary image, which means each pixel of the image is either black or white. This is necessary because if we classify background and object(inner side) with black and white, it becomes easy to convert them to real video images.

TP1-3: INTERIM REPORT

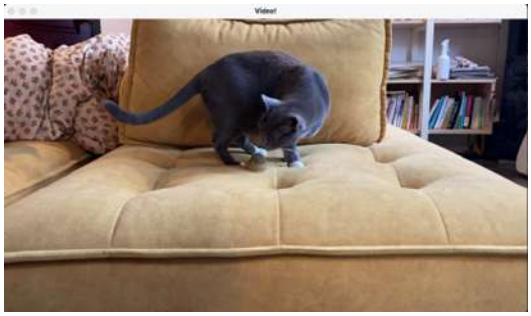
We tried using the background estimation using median value and contouring method to create a mask on the moving object in the video. Our experiments show that image subtraction of each frame from the calculated background image gives better results. Follow are the final output results from both the methods:

(img 1: mask created using image subtraction method)

(img 2: mask created using contouring method)

Method 1 : Finding median frame and using mask

1. **Import foreground video & background image sources**



| Extract cat from this video

| Background image

2. **Convert color of foreground image, RGB to GRAYSCALE**

```
frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

3. **Setting the threshold**

```
threshold, mask = cv2.threshold(dif_frame, 1, 255,cv2.THRESH_BINARY)
```

-) set the threshold for deciding whether to put the pixel in foreground more than threshold value or background (less than threshold value)

4. **Make mask from the foreground video**

```
mask[mask>0]=255
#diff = diff.astype(np.uint8)

#inverting the mask
inverted_mask = cv2.bitwise_not(mask)
```

5. **Create new foreground video & background video with the mask**

6. **Combine 2 videos**

```
finalvideo = cv2.bitwise_or(foreground_vid,background_vid)
```

final result video

Method 2 : Contouring

>Using this technique, the object is distinguished by drawing lines of objects. However, if the background is not monochrome, there is a problem that we cannot clearly distinguish between the object we want to separate and the background image. When we put chroma-key images with green backgrounds in this algorithm, it is well distinguished, but when we put the images we took, we can see that the background images continue to be applied to the cat.

1) **inverse image**

<when using green background>

<when using video we took>

2) mask imag

<when using green background>

<when using video we took>

> Process of the algorithm

1. **inverse the mask**

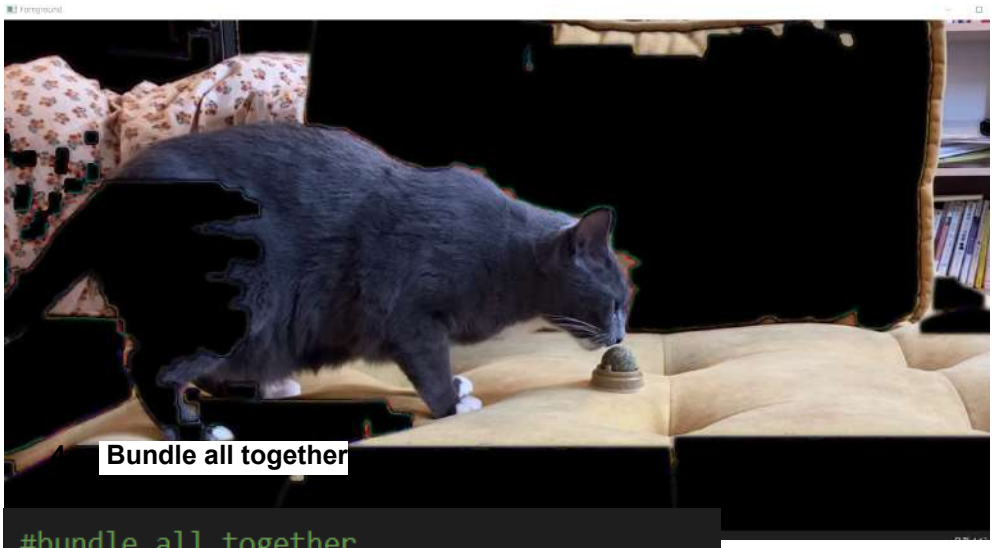
```
# #inverse mask
inverse=cv2.bitwise_xor(mask,inverse)
```

2. **mask the background image**

```
#mask backgrond image
inverse=cv2.bitwise_and(inverse, img)
```

3. **Blend video we took and background image**

```
#blend movie and foreground image
mask=cv2.bitwise_and(mask,frame)
```




```
#bundle all together
mask=cv2.bitwise_or(mask,inverse)
```

REQUIRED WORK IN THE FUTURE:

1. **Create an if statement to set different threshold values for videos that have a GREEN SCREEN and videos that are without any special colored background.** From our experiments, we found that if the threshold is set to 1, the video with green screen is masked pretty well. But, on other videos that have more complicated background, a threshold of 25-30 works better. So, we plan to create a if loop where if the pixel values of the background lie in the HSV range of green color, the threshold will be set to 1. If not, it will be set to 25.

2. Add code lines in the beginning of the code so that the user can **input their input video and image file path name**

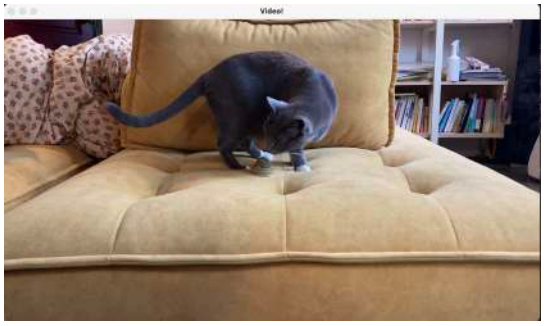
Final Code:

1. **Set up the environment**
   - Import the necessary libraries
   - Get user input for video and the new background
   - Check if the video path and image path exists in the system

```
import cv2
import numpy as np
import os

videopath = input("Enter video path name: ")
imagepath = input("Enter new background image path name: ")


if os.path.exists(videopath):
    if os.path.exists(imagepath):
```

2. **Getting the video information and resizing and recoloring the background image:**
   - Get video attributes
   - resize background image to size of video
   - change background image to grayscale

```
if os.path.exists(videopath):
    if os.path.exists(imagepath):

        stream = cv2.VideoCapture(videopath)
        fps = stream.get(cv2.CAP_PROP_FPS)
        nframe = int(stream.get(cv2.CAP_PROP_FRAME_COUNT))
        width = int(stream.get(3))
        height = int(stream.get(4))

        #converting the image to grayscale
        bgimg = cv2.imread(imagepath)
        bgimg = cv2.resize(bgimg,[width,height])

        gray_bgimg = cv2.cvtColor(bgimg,cv2.COLOR_BGR2GRAY)
```

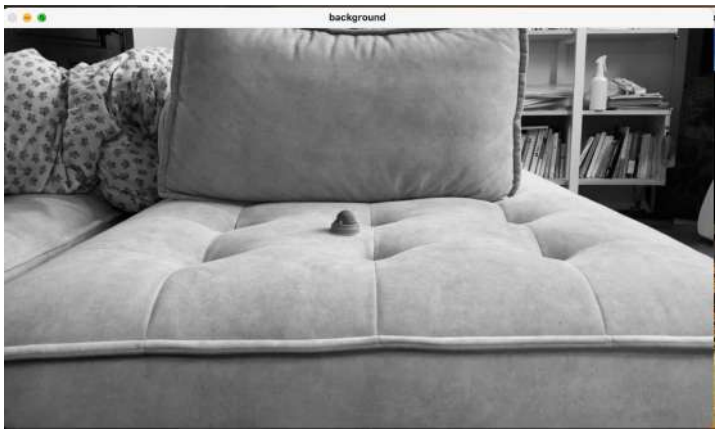


Original video screenshot

New background

3. **Calculating the video background using the median values of randomly selected frames from the video:**
   - Read the video
   - Append random frames from the video to the empty frame list
   - calculate the median value of the frames —> this is our original background
   - convert original background to grayscale and blur it to remove slight differences in pixel values between each frame

```
    #creating variables to adjust the noise in the mask later
    dilate_iter = 8
    erode_iter = 8
    threshold = 25

    #if there is no video file, it will exit the program
    if not stream.isOpened():
        print("No stream :(")
        exit()

    #create an empty list for getting the pixel values for the selected frame
    frames = []

    '''without the stream.get in the front, cv2.CAP_PROP_POS_FRAMES gives 7 as output
        #and this somehow is the background image. I am not sure how it works. I wrote this code by mistake'''

    for fid in range(cv2.CAP_PROP_POS_FRAMES):
        stream.set(cv2.CAP_PROP_POS_FRAMES, fid)
        ret, frame = stream.read() #reading each frame and getting the frame pixel information
        if not ret: # if no frames are returned, then exit the loop
            print("SOMETHING WENT WRONG")
            exit()
        frames.append(frame) #append the frame in frames list made above

    #The median frame here is our background
    median = np.median(frames, axis=0).astype(np.uint8) #find the median value of every pixel. Median value = background
    median = cv2.cvtColor(median, cv2.COLOR_BGR2GRAY) #convert median value to grayscale
    median = cv2.GaussianBlur(median, (5,5), 0) #blur to reduce differences in pixel values in consequent frames
```



Extracted background from video

4. **Create a recorder to download the final video with new background**
   - Creating the recorder
   - setting the frame to 0 again (it was set to some other frame number while finding the median)

```
    #recorder
    fourcc = cv2.VideoWriter_fourcc('m', 'p', '4', 'v')
    recorder = cv2.VideoWriter('no_background.mp4', fourcc, fps, (width,height))

    #setting the stream back to 0 frame
    stream.set(cv2.CAP_PROP_POS_FRAMES, 0)
```

5. Reading each frame and creating the foreground masks and background masks:

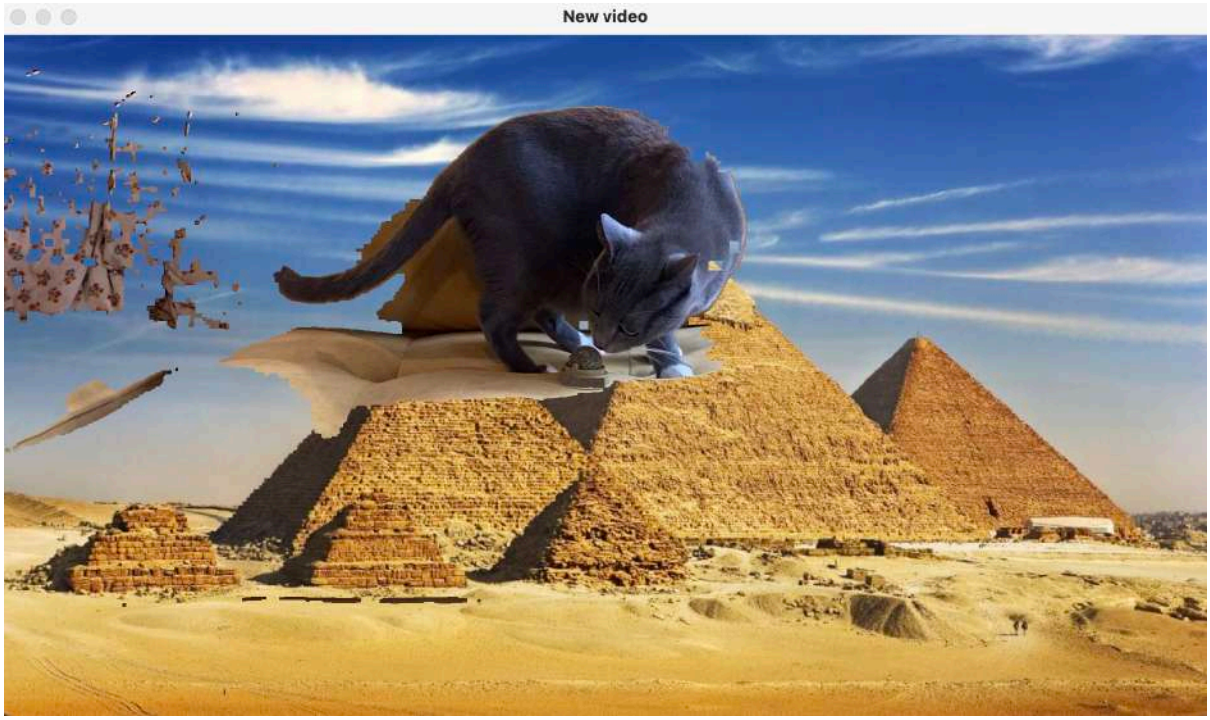
```
60          while True:
61              #reading the video file
62              ret, frame = stream.read()
63
64              #print(bgimg.shape, frame.shape)
65              if not ret: # if no frames are returned
66                  print("No more stream :(")
67                  break
68
69              '''subtracting background image from every frame by using absolute difference'''
70              #coverting frame to grayscale
71              mask_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
72              #applying Gaussian Blur to remove noise
73              mask_frame = cv2.GaussianBlur(mask_frame, (5,5), 0) #converting the video into grayscale
74              #getting the absolute difference value between median frame (background) and each frame
75              dif_frame = cv2.absdiff(median, mask_frame)
76
77              '''setting the threshold for deciding whether to put the pixel in foreground
78              more than threshold value or background (less than threshold value)'''
79
80              threshold, mask = cv2.threshold(dif_frame, threshold, 255,cv2.THRESH_BINARY)
81
82              #retouching the mask to remove holes and noise
83              mask = cv2.dilate(mask, None, iterations=dilate_iter)
84              mask = cv2.erode(mask, None, iterations=erode_iter)
85              mask = cv2.GaussianBlur(mask, (5, 5), 0)
86
87              mask[mask>0]=255
88
```

Foreground mask

Background mask

6. Applying mask to the foreground video and the new background image & combining both the videos

```
            inverted_mask = cv2.bitwise_not(mask)

            #creating foreground and background videos with the mask and inverted mask respectively
            foreground_vid = cv2.bitwise_and(frame,frame, mask=mask)
            background_vid = cv2.bitwise_and(bgimg, bgimg, mask = inverted_mask)

            #combining the new foreground and background videos
            finalvideo = cv2.bitwise_or(foreground_vid,background_vid)

            cv2.imshow('New video', finalvideo)
            cv2.imshow("Video!", frame)
            cv2.imshow("background", median)
            cv2.imshow("foreground with mask", foreground_vid)
            cv2.imshow("new background with mask",background_vid)

            recorder.write(finalvideo)

            cv2.waitKey(15)
            if cv2.waitKey(1) == ord('q'): # press "q" to quit
                break

        #cleaning up
        stream.release()
        cv2.destroyAllWindows()
    recorder.release()

else:
    print("Please enter the correct path name :( ")
```



Video with new background (screenshot)

Further development scope and problems:

1. **Different threshold for chromakey videos:** For videos with green screen or any other color screen, the threshold for image subtraction works better when its set to 1 instead of 25 (threshold for normal videos). So, maybe the code could be developed to add that a condition where it identifies the pixel hav values of the detected background and if it lies in the range of green or any other color hav range, then the threshold is set to 1, if not it is set to 25.
2. **Result is not the best:** As you can see in the final result, the video is not the cleanest. There is a lot of noise in in other parts of the video. Maybe, we could add some code lines to detect contours in the mask that are smaller than a certain size and remove them. This way only the biggest main subject will be in the mask.