Assignment 0
Chroma Keying

Arpit Gupta
2018201048

Problem Statement

Chroma keying is a technique used for combining two frames or images by replacing a color or a color range in one frame with that from the another frame. It can be used on two images as well as two videos.

This technique is also referred by various terms depending on the color used in the background. Chroma keying can be done with backgrounds of any color that are uniform and distinct, but green and blue backgrounds are more commonly used.

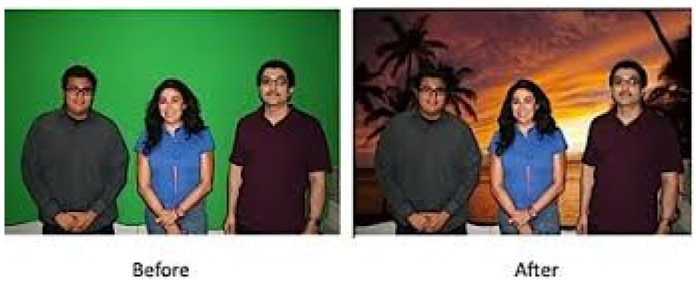

Fig.1 – An example of chroma keying with green color

Solution

First step of the solution includes differentiating between foreground and background pixels. Once this is achieved, we can then easily replace every background pixel with the corresponding pixel of the desired background.

For differentiating, range of the color which needs to be removed/replaced that is background color is taken as input from the user. Using this range of [B,G,R] values a mask of the frame/image is created. This mask marks all the foreground pixels as black and background pixels as white.

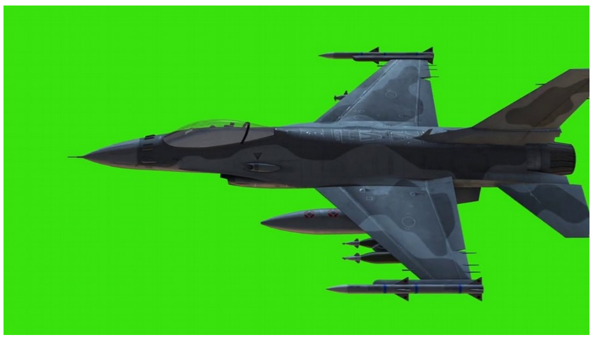

Fig.2 – Original Image

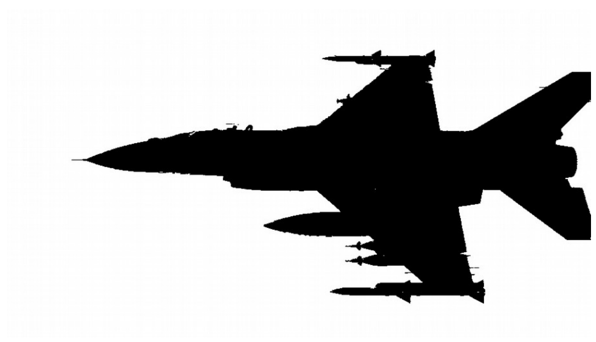

Fig.3 – Mask generated

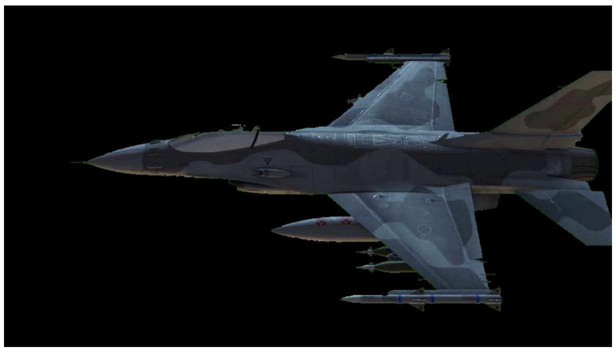

Fig.4 – New image generated using original and mask

Using the same given range of [B,G,R] values the corresponding background image is masked in opposite way that is foreground pixels (of original image) are marked as white in this background image and background pixels remain same.

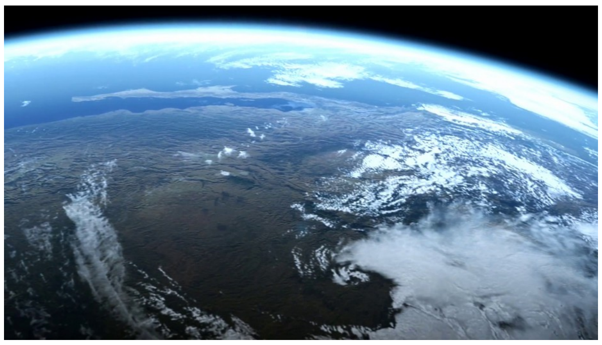
Fig.5 – Original Background frame/image

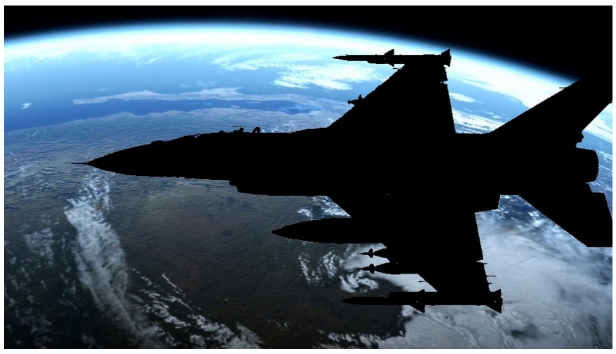
Fig.6 – Masked Background image

Now to get the final frame both the images that is masked foreground image (fig.4) and masked background image (fig.6) are added.

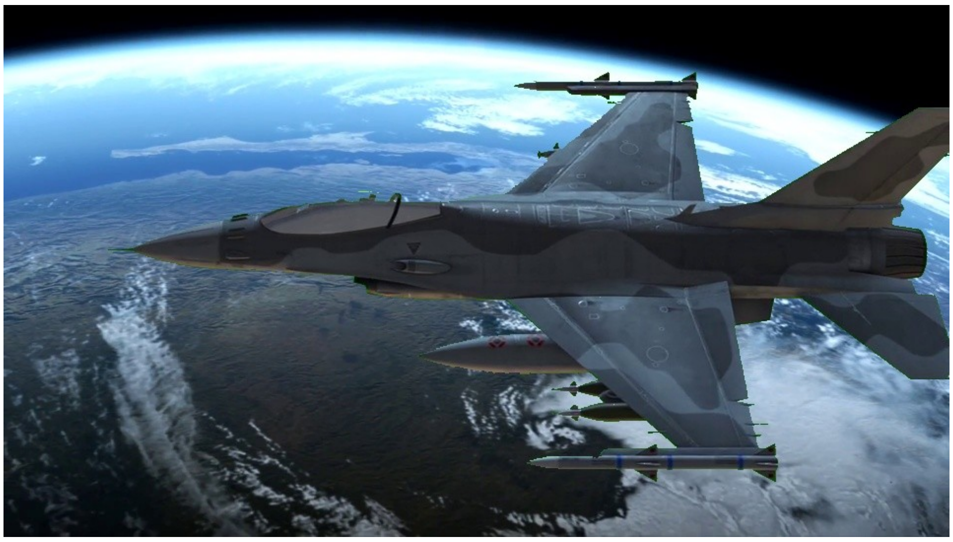
Fig.7 – Final image after chroma keying

Learning and Experiments

This assignment helped me to understand a lot about how the images and videos can be manipulated and edited for fulfilling our requirements. For example, the film industry uses this technique so frequently in order to maximize our entertainment.

I learned to read/write images and videos using opencv library in-built functions. I became familiar with *Mask* of an image and how useful it is.

For identifying background and foreground pixels, I *experimented* with various in-built algorithms of opencv which includes -

1. *Grabcut method*
2. *Background subtraction method*

The matrix operations can be performed with so much ease using opencv in-built functions like *cv2.inRange()*, *cv2.resize()*, etc. For example -

I needed to iterate over the entire frame and replace each background pixel with black. Initially I used loops which made my script very slow as I needed to do this on each and every frame of the video. But, *cv2.inRange()* function made this operation very fast.

Code

Part 1 : Link for the video used.

Convert a given video to its constituent images.

```
import numpy as np
import cv2
import os

cap = cv2.VideoCapture("bg2.mov") #creating a VideoCapture object
i=1
path = os.getcwd();
os.makedirs("images", exist_ok=True) #creating a new folder to store images
os.chdir(path+"/images")

while(cap.isOpened()):
    ret, frame = cap.read() # reading current frame from video
    if ret==True:
        cv2.imshow('frame',frame) # displaying it
        cv2.imwrite(str(i)+'.jpg',frame) # storing the frame as a jpg image
        i = i+1
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()
cv2.destroyAllWindows()
```

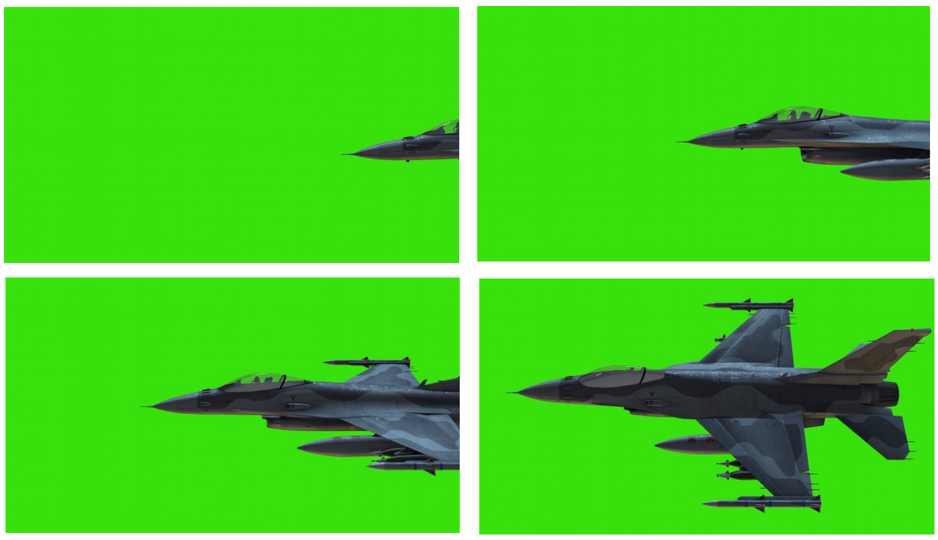

Fig.8 - Some of the output images

Merge a set of images in a folder into a single video.

Takes frame rate for the output video as argument. If not given then fps is set to 20.

```
import numpy as np
import cv2
import os
import sys

path = os.getcwd();
os.makedirs("images", exist_ok=True)
os.chdir(path+"/images")

if(len(sys.argv)==2):
        fps = int(sys.argv[1])
else:
        fps = 20

fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('im_to_vid.avi',fourcc, fps, (640,480)) # creating VideoWriter object to save the output video

i=1

while True:
        img = cv2.imread(str(i)+'.jpg',1)
        i = i+1
        if img is not None:
                img = cv2.resize(img,(640, 480),interpolation = cv2.INTER_AREA)
                cv2.imshow('image',img)
                out.write(img)
                if cv2.waitKey(1) & 0xFF == ord('q'):
                        break
        else:
                break

out.release()
cv2.destroyAllWindows()
```

Link for the output video.

Part 2 : **Converting a live video being captured from web camera of my latop to images**

```
import numpy as np
import cv2
import os

cap = cv2.VideoCapture(0) #creating a VideoCapture object
i=1
path = os.getcwd();
os.makedirs("images", exist_ok=True) #creating a new folder to store images
os.chdir(path+"/images")

while(cap.isOpened()):
   ret, frame = cap.read() # reading current frame from video
   if ret==True:
         cv2.imshow('frame',frame) # displaying it
         cv2.imwrite(str(i)+'.jpg',frame) # storing the frame as a jpg image
         i = i+1
         if cv2.waitKey(1) & 0xFF == ord('q'):
                 break
   else:
         break

cap.release()
cv2.destroyAllWindows()
```

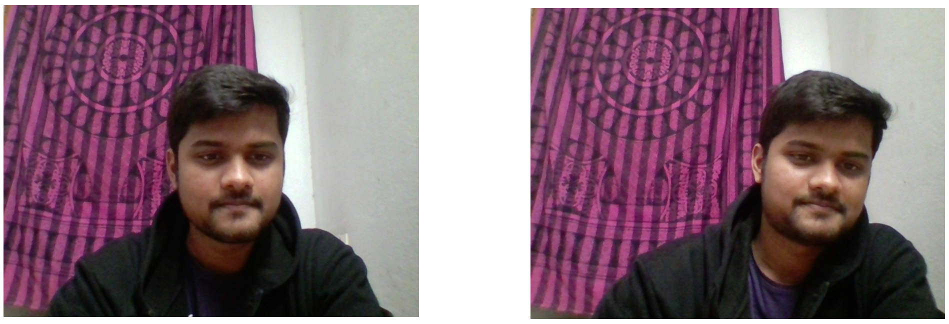

Fig.9 - Some of the output images

Part 3 : **Chroma Keying**

```
import numpy as np
import cv2
import os

cap = cv2.VideoCapture("f.mp4") # creating VideCapture object for the foreground video
back = cv2.VideoCapture("bg2.mov") # creating VideCapture object for the background video

fps = cap.get(cv2.CAP_PROP_FPS) # calculating frame rate of the foreground video

fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi',fourcc, fps, (640,480)) # creating VideoWriter object to save the
output video

while(cap.isOpened() and back.isOpened()):
    ret, img = cap.read() # reading frame of the foreground video
    re, bg = back.read() # reading frame of the background video
    if ret==True:
        bg = cv2.resize(bg,(640, 480),interpolation = cv2.INTER_AREA)
        img = cv2.resize(img,(640, 480),interpolation = cv2.INTER_AREA)
        lower_green = np.array([0,110,0]) # setting lower range of [B,G,R] values
        upper_green = np.array([120,280,120]) # setting upper range of [B,G,R] values
        mask = cv2.inRange(img, lower_green, upper_green) # generating mask of the foreground
frame
        img[mask != 0] = [0, 0, 0] # applying mask to foreground frame
        bg[mask == 0] = [0,0,0] # applying mask to background frame but in opposite way
        final = bg + img # creating final image
        cv2.imshow('frame',final)
        out.write(final)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```

Link for input video 1.
Link for input video 2.
Link for the output video.