Report: Assignment - 0

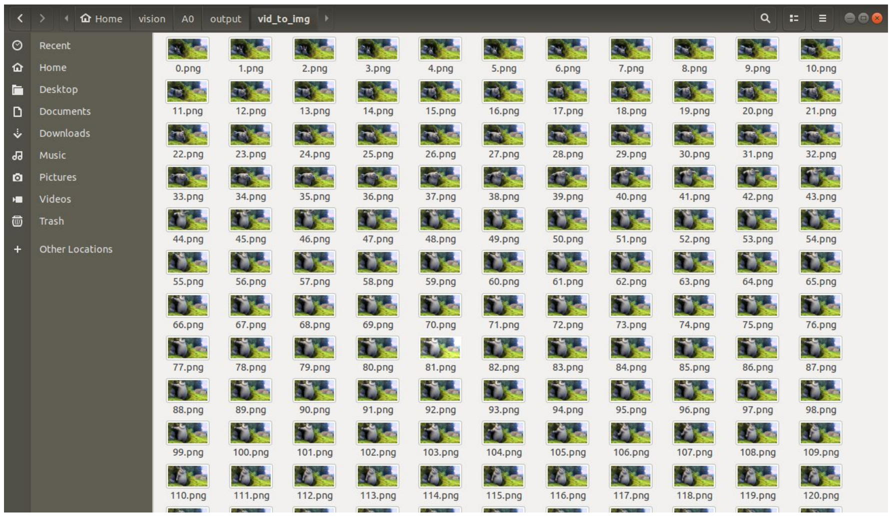
Chris Andrew
2018701019

- All the input data used in this assignment has been zipped into a folder called **data** and can be downloaded here.
- All the results generated in this assignment have been zipped into a folder called **output** and can be downloaded here.

---

1. Conversion between videos and images:

- **Problem definition:** Write a program to convert a given video to its constituent images. Your output should be in a specified folder. Write another program that will merge a set of images in a folder into a single video. You should be able to control the frame rate in the video that is created.
- **Challenges faced:**
  - Compiling OpenCV from source.
  - Testing the video module in OpenCV.
  - Reading tutorials on using the video module in OpenCV.
- **Results:**
  - I took a sample video(**sample.mp4** in the **data** folder) and converted the entire video into frames.
  - These frames are then stored in an output folder(parameter) as numbered frames starting from 0. (**vid_to_img/** folder in **output** folder)


Home
vision
A0
output
vid_to_img
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
0.png 1.png 2.png 3.png 4.png 5.png 6.png 7.png 8.png 9.png 10.png
11.png 12.png 13.png 14.png 15.png 16.png 17.png 18.png 19.png 20.png 21.png
22.png 23.png 24.png 25.png 26.png 27.png 28.png 29.png 30.png 31.png 32.png
33.png 34.png 35.png 36.png 37.png 38.png 39.png 40.png 41.png 42.png 43.png
44.png 45.png 46.png 47.png 48.png 49.png 50.png 51.png 52.png 53.png 54.png
55.png 56.png 57.png 58.png 59.png 60.png 61.png 62.png 63.png 64.png 65.png
66.png 67.png 68.png 69.png 70.png 71.png 72.png 73.png 74.png 75.png 76.png
77.png 78.png 79.png 80.png 81.png 82.png 83.png 84.png 85.png 86.png 87.png
88.png 89.png 90.png 91.png 92.png 93.png 94.png 95.png 96.png 97.png 98.png
99.png 100.png 101.png 102.png 103.png 104.png 105.png 106.png 107.png 108.png 109.png
110.png 111.png 112.png 113.png 114.png 115.png 116.png 117.png 118.png 119.png 120.png

- For reconstruction of video from frames, we give a list of number frame images(starting from 0 to N) where the k^th frame is named as k.* where * is the image codec extension. We store these in a folder and pass the folder path as a parameter for the reconstruction. One such sample is there in **frames/** folder in the data folder.
- The video is reconstructed using a given frame rate(fps) which is also a parameter to control the speed of the video. The reconstructed video is stored in a specified output file.(**output.avi** in the **output** folder)

Results for these experiments can be found in the output folder.

- **Code(video to images)**

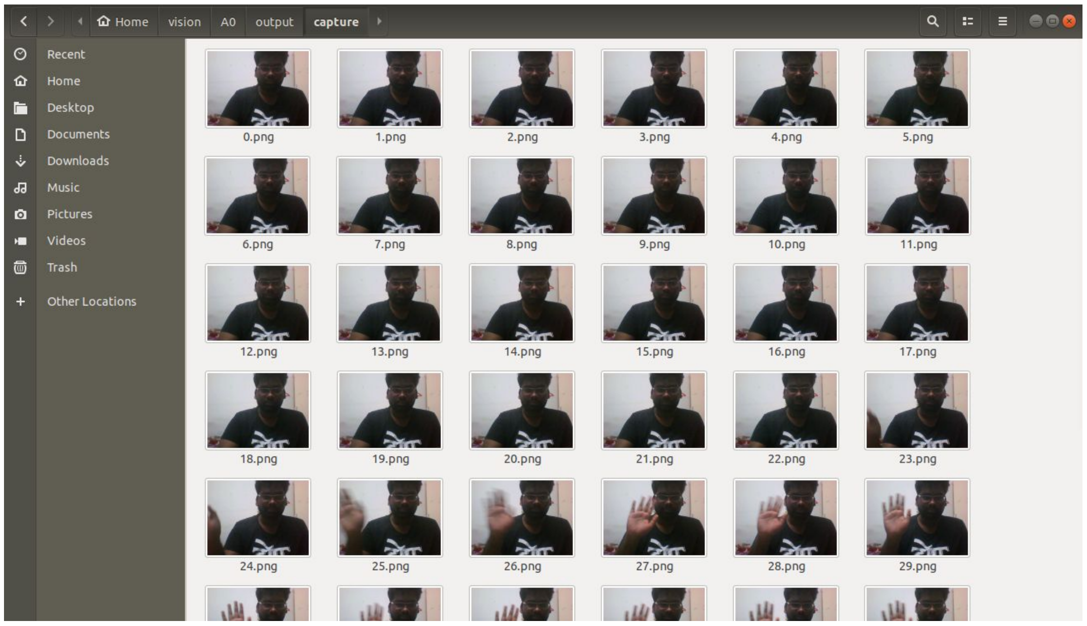
```
import cv2
import os

def convert_to_img(vid_file, output_folder):
    """Converts a video to a series of images and saves to a specified
location."""
    cam = cv2.VideoCapture(vid_file)
    counter = 0
    ret = True
    while(ret):
        ret, frame = cam.read()
        if not ret:
            break

        cv2.imshow('frame', frame)
        cv2.imwrite(os.path.join(output_folder, str(counter) + ".png"),
frame)
        counter += 1
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cam.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    video_file = "data/sample.mp4"
    output_folder = "output/vid_to_img/"
    convert_to_img(video_file, output_folder)
```

- **<u>Code(images to video)</u>**

```
import cv2
import os


def convert_to_vid(frames_folder, video_file, fps=25, frame_size=(1280,
720)):
    """Converts a series of images to video and saves to a specified
location."""
    codec = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter(video_file, codec, float(fps), frame_size,
isColor=True)

    frames = os.listdir(frames_folder)
    filetype = "." + frames[0].split(".")[1]

    frames = [x.split(".")[0] for x in frames]
    frames.sort(key=int)
    frames = [x + filetype for x in frames]

    for f in frames:
        frame = cv2.imread(os.path.join(frames_folder, f))
        out.write(frame)

    out.release()
    cv2.destroyAllWindows()


if __name__ == "__main__":
    frames_folder = "data/frames/"
    video_file = "output/output.avi"
    fps = 10
    frame_size = (1280, 720)

    convert_to_vid(frames_folder, video_file, fps=fps,
frame_size=frame_size)
```

2. Capturing video from camera.

- **Problem definition:** Learn how to capture frames from a webcam connected to your computer and save them as images in a folder. You may use either the built-in camera of your laptop or an external one connected through USB. You should also be able to display the frames (the video) on the screen while capturing.
- **Challenges faced:**
  - Compiling OpenCV from source.
  - Testing the video module in OpenCV.
  - Reading tutorials on using the video module in OpenCV.
- **Results:**
  - I captured a video of myself using the webcam in my laptop and saved them as frames in the **capture/** folder in **output** folder.
  - These frames are stored in an output folder(parameter) as numbered frames starting from 0.

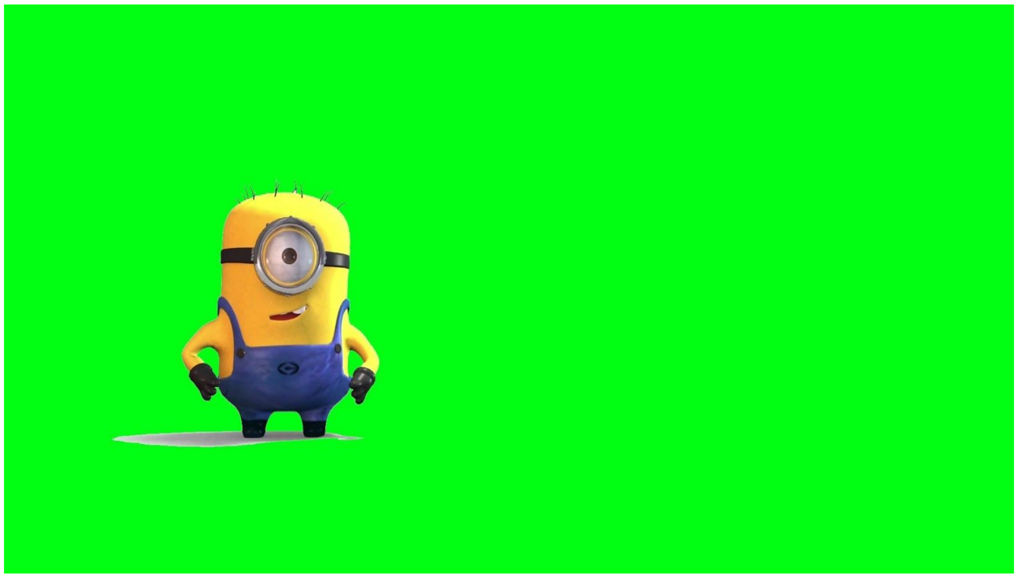


Results for these experiments can be found in the output folder.

- **Code:**

```python
import cv2
import os


def capture_video(output_folder):
    """Capture frames from camera and saves to a specified 
location."""
    cap = cv2.VideoCapture(0)
    counter = 0

    while(True):
        ret, frame = cap.read()

        cv2.imshow('frame', frame)
        cv2.imwrite(os.path.join(output_folder, str(counter) + 
".png"), frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()


if __name__ == "__main__":
    output_folder = "output/capture/"

    capture_video(output_folder)
```

3. Chroma Keying:

- **Problem definition:** Chroma key compositing, or chroma keying, is a visual effects/post-production technique for compositing (layering) two images or video streams together based on color hues (chroma range). Usually there is a green/blue screen in a stream/image which is replaced with another stream/image from a different source. This is also called a green screen image and is recorded/shot with a green background that is later replaced with some other content using chroma keying.
  Here is a sample of a green screen image:
  
  The green part of the image is later replaced with some other content.

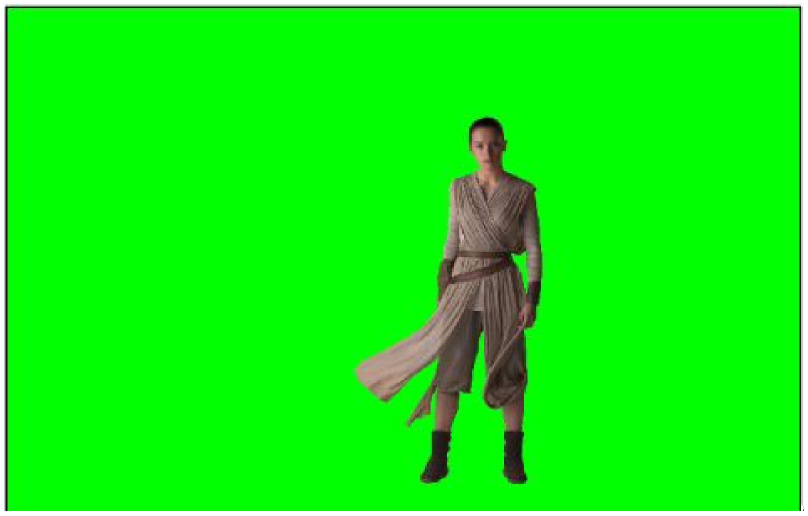
- **Challenges faced:**
  - The main problem in Chroma Keying is to create a binary map that separates the foreground pixels(character) from the background(green screen) once this map is created we can combine the two image by multiplying with the map.
  - For map creation, we need to first specify the color of the background, this is the keycolor. Once we know the key color, we try and identify which pixel is a keycolor and which is not.
  - I compared the Euclidian distance from the pixel color to the keycolor. Pixels that are below a certain distance threshold are classified as background pixels and the remaining are classified as foreground.
  - The threshold parameter needs to be manually set as different videos will have different quality of the green screen(uniform or uneven colored).

- The keycolor also needs to be set, in our experiments we used green screens and so the key color is set to BGR green (0, 255, 0). If a blue screen is used, it needs to be set to BGR blue (255, 0, 0). Note that OpenCV uses the BGR color scheme rather than the commonly used RGB scheme.

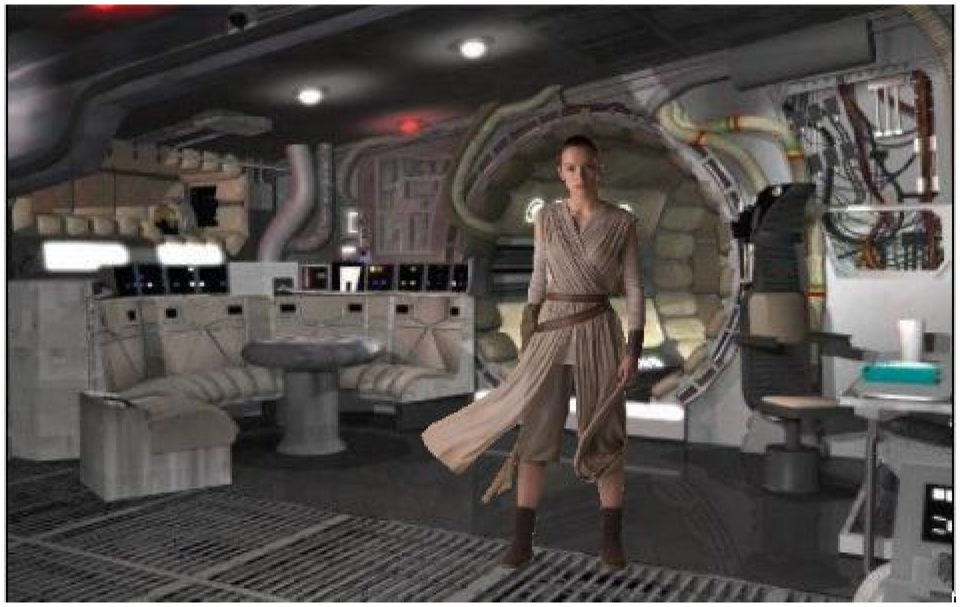
- **Results:**
    - I first tried chroma keying with a few images to test if it worked before working on videos. Here is one set of images that I tested(distance threshold is set to 200)

      Background:

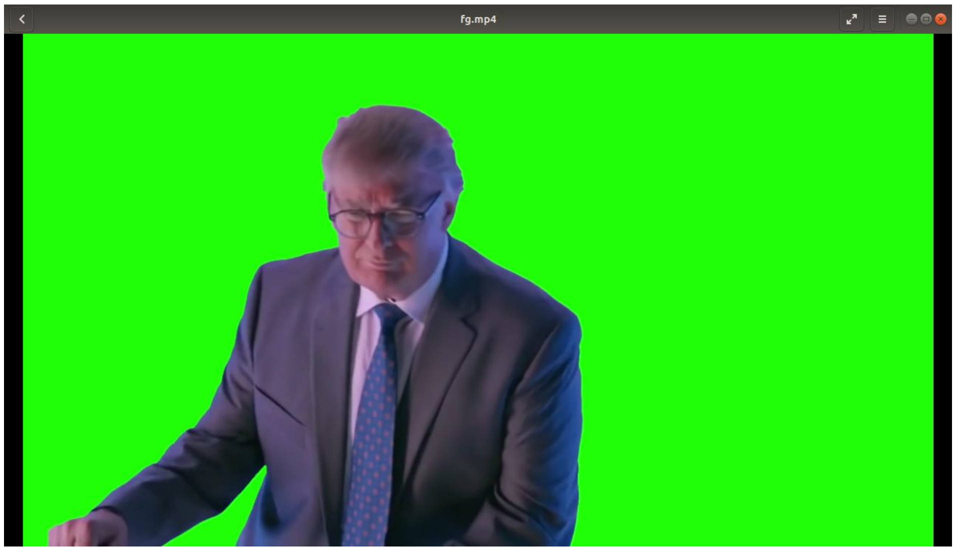
      Foreground:

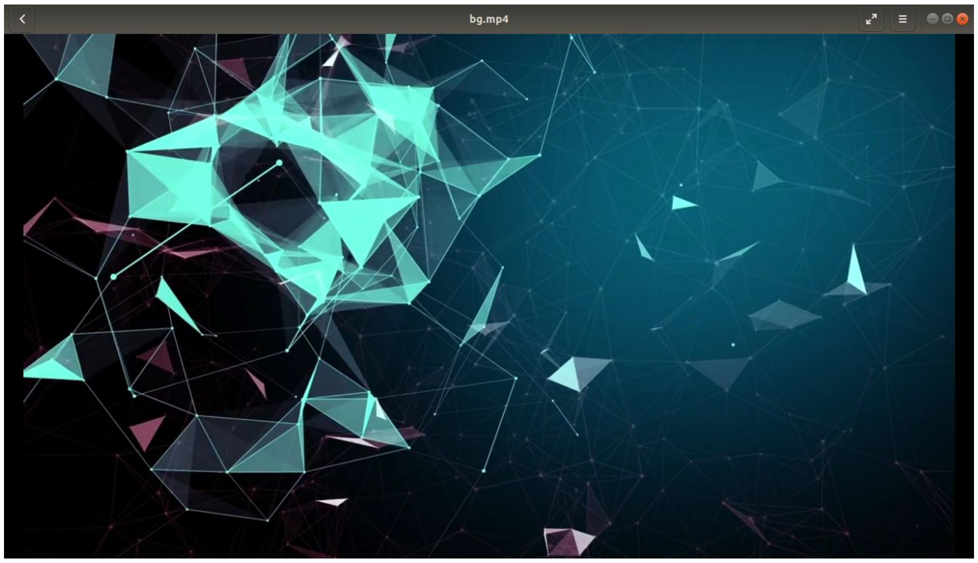
      

Result:



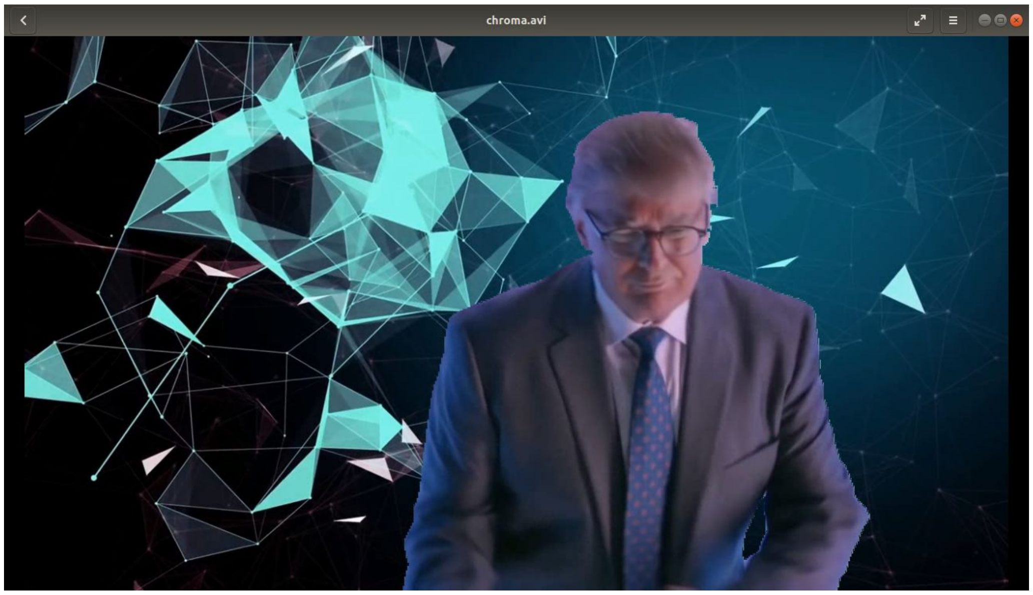
- Once I could get the images to combine, I then moved on to videos. For videos, I selected one of Donald Trump dancing in front of a green screen along with a funky music video background.





These are stored as **fg.mp4** and **bg.mp4** in the **data** folder.

Here is the result:



chroma.avi in the **output** folder.

- **Code:**

```
import cv2
import numpy as np


def get_frames(vid_file, title):
    """Converts a video to a series of images and saves to a
specified location."""
    cam = cv2.VideoCapture(vid_file)
    frames = list()
    ret = True
    while(ret):
        ret, frame = cam.read()
        if not ret:
            break
        frames.append(frame)

    cam.release()
    cv2.destroyAllWindows()
    return frames


def convert_to_vid(frames, video_file, fps=25, frame_size=(1280,
720)):
    """Converts a series of images to video and saves to a specified
location."""
    codec = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter(video_file, codec, float(fps), frame_size,
isColor=True)

    for frame in frames:
        out.write(frame)

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

```python
def ChromaKey(fg, bg, keyColor, threshold=200):
    """Replace keyColor in fg with bg."""
    color_diff = fg.astype(np.float) - keyColor.astype(np.float)
    color_dist = np.sqrt(np.sum(color_diff * color_diff, axis=2))
    color_pixels = (color_dist > threshold).nonzero()

    transform_map = np.zeros(fg.shape, dtype=np.uint8)
    transform_map[color_pixels] = 1
    inverse_map = 1 - transform_map
    transform_map = transform_map * fg
    inverse_map = inverse_map * bg
    final_image = transform_map + inverse_map

    return final_image

def compose(fg_vid, bg_vid, keyColor, output_vid):
    print("Getting frames from FG video")
    fg_frames = get_frames(fg_vid, "Foreground video")
    print("Getting frames from BG video")
    bg_frames = get_frames(bg_vid, "Background video")

    bg_len = len(bg_frames)
    output = list()

    print("Chroma keying two videos")
    for i, f in enumerate(fg_frames):
        out = ChromaKey(f, bg_frames[i % bg_len], keyColor)
        output.append(out)
    print("Saving video at", output_vid)
    convert_to_vid(output, output_vid)

if __name__ == "__main__":
    output_vid = "output/chroma.avi"
    fg_vid = "data/fg.mp4"
    bg_vid = "data/bg.mp4"
    keyColor = np.array([0, 255, 0], dtype=np.uint8)
    compose(fg_vid, bg_vid, keyColor, output_vid)
```