

# CS7.505 - Assignment 0

## OpenCV and Chroma Keying

Avneesh Mishra  
 avneesh.mishra@research.iiit.ac.in \*

March 6, 2022

## Contents

<b>1</b>	<b>Installing and Testing OpenCV</b>	<b>2</b>
1.1	OpenCV Setup . . . . .	2
1.2	Testing Installation of OpenCV . . . . .	2
<b>2</b>	<b>Chroma Keying with OpenCV</b>	<b>4</b>
2.1	Video and Images . . . . .	4
2.1.1	Video to images . . . . .	4
2.1.2	Images to Video . . . . .	5
2.2	Capture Images through Webcam . . . . .	6
2.3	Chroma Keying . . . . .	8
<b>3</b>	<b>Additional Tasks</b>	<b>11</b>
3.1	Shot Detection . . . . .	11
3.2	Face Detection . . . . .	11

## List of Figures

1	Test images . . . . .	3
2	Sequence of images . . . . .	5
3	Saved videos . . . . .	7
4	Images captured through webcam . . . . .	8
5	Chroma keying results . . . . .	11

## Listings

..../python/check_cv2_version.py . . . . .	2
1 Transform images . . . . .	3
2 vid_to_imgs.py . . . . .	4
3 imgs_to_vid.py . . . . .	6
4 webcam_capture.py . . . . .	7
5 chroma_keying.py . . . . .	9

**Shared results** The results and code presented here are available through this link<sup>1</sup> (only IIITH access). The author can be contacted for the material too. Most of the items are included here as code listings and figures.

---

\*M.S. by research - CSE, IIIT Hyderabad, Roll No: 2021701032

<sup>1</sup><https://bit.ly/3A3MC12>

# 1 Installing and Testing OpenCV

Main docs: <https://docs.opencv.org/4.x/>

Python docs: [https://docs.opencv.org/4.x/d6/d00/tutorial\\_py\\_root.html](https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html)

## 1.1 OpenCV Setup

The procedure to setup OpenCV using Anaconda on windows is listed below

1. Download and install Anaconda from here.
2. Setup the anaconda environment for the shell using `conda init`
3. Setup the environment  
Create the environment

```
conda create -yn "cv-cs7-505"
conda activate cv-cs7-505
```

Install Python 3.9 into it

```
conda install python=3.9
```

Install the essential packages (before OpenCV)

```
conda install numpy jupyterlab
```

4. Install OpenCV using pip

Install using pip

```
pip install opencv-python opencv-contrib-python
```

## 1.2 Testing Installation of OpenCV

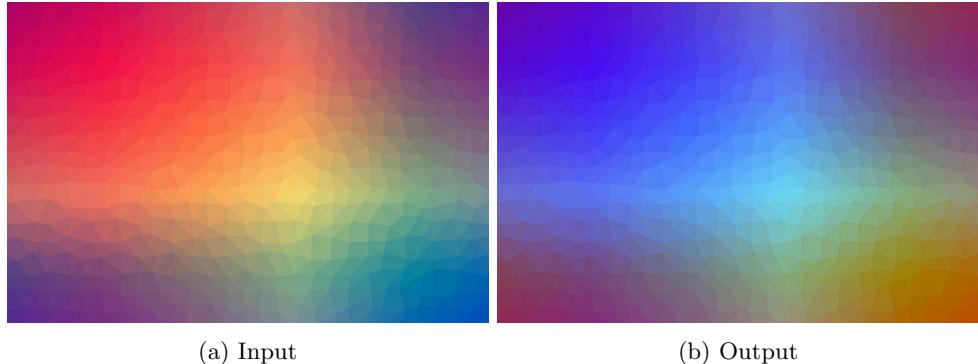
### Checking Version

Check version using the script below

```
1 # Check OpenCV version
2 # %% Import everything
3 import cv2 as cv
4 import numpy as np
5
6 # %% Main entrypoint
7 if __name__ == "__main__":
8     print(f"OpenCV version: {cv.__version__}")
9     print(f"Numpy version: {np.__version__}")
10
11 # %%
```

The output of the above script is

```
OpenCV version: 4.5.5
Numpy version: 1.21.2
```



(a) Input

(b) Output

Figure 1: Test images

The red and blue channels are flipped (their layer intensities are swapped). Check code listing 1

### Reading and changing color channels

Consider the images in figure 1. The code to get this output is given below

```

1 # Change the color channels of an image
2 """
3     Flips the Red and Blue channel of an image
4 """
5
6 # %% Import everything
7 import cv2 as cv
8 import numpy as np
9
10 # %% Functions
11 # Show image in a window
12 def show_img(img, win_name = "Image"):
13     cv.namedWindow(win_name, cv.WINDOW_GUI_EXPANDED)
14     while True:
15         cv.imshow(win_name, img)
16         if cv.waitKey(1) == ord('q'):
17             break
18     cv.destroyAllWindows()
19
20 # %% Main module
21 if __name__ == "__main__":
22     # Read image
23     img_in = cv.imread("./images/test.jpg")
24     # Show image
25     show_img(img_in, "Input")
26     img_out = img_in[:, :, ::-1]
27     show_img(img_out, "Output")
28     # Save output
29     cv.imwrite("./images/output.jpg", img_out)
30
31 # %%

```

Listing 1: Transform images

## 2 Chroma Keying with OpenCV

### 2.1 Video and Images

#### 2.1.1 Video to images

**Problem** Given a video, convert it into constituent images

**Experiments & Learning** Experiments performed are listed below

1. Giving user input to the program: Tried using the `input` built-in to read from console. Finally settled at using argparse. Learned how to parse parameters professionally using argparse.
2. Using `VideoCapture` and `read` functions to fetch images from a video file, and writing images to disk using `imwrite`. Some experiments were run where the output prefix directory did not exist, so added the code to create the directory first.

**Solution** Download video from here and store as `./videos/vtest.avi`. Then run the following

```
python .\vid_to_imgs.py -n 10
```

The code is shown in listing 2. Output is in figure 2.

```
1 # Convert a video to images
2 """
3     Given a video, generate the images. Run as main.
4 """
5
6 # %% Import everything
7 import cv2 as cv
8 import numpy as np
9 import argparse
10 import sys
11 import os
12
13 # %% Argument parser
14 parser = argparse.ArgumentParser(
15     formatter_class=argparse.ArgumentDefaultsHelpFormatter)
16 parser.add_argument('-i', '--vid-file', default='./videos/vtest.avi',
17     help="Video file to read")
18 parser.add_argument('-n', '--num-imgs', default=0, type=int,
19     help="The maximum number of images to output from video (0=all)")
20 parser.add_argument('-o', '--out-prefix', default='./out/img',
21     type=str, help="Output prefix for images")
22
23 # %% Main entrypoint
24 if __name__ == "__main__":
25     # Parse all (known) arguments
26     args, unknown_args = parser.parse_known_args(sys.argv)
27     # Check if output directory (if passed) exists
28     out_dir = os.path.split(args.out_prefix)[0]
29     if not os.path.isdir(out_dir):
30         print(f"Folder {out_dir} is being created")
31         os.makedirs(out_dir)
32     # Read video from file
33     cap = cv.VideoCapture(args.vid_file)
34     try:
35         # Read frames
36         fnum = 0
37         while cap.isOpened():
38             ret, frame = cap.read()
39             if not ret:
40                 print("Probably EOF reached!")
41                 break
42             cv.imshow("Video Feed", frame)
43             if cv.waitKey(100) == ord('q'):
44                 print(f"Break encountered")
45                 break
46             if args.num_imgs == 0 or fnum < args.num_imgs:
47                 # Write to disk
```



(a) Image 1

(b) Image 5

(c) Image 10

Figure 2: Sequence of images

First, fifth, and tenth image in the 10-image sequence produced by listing 2.

```

48         cv.imwrite(f"{args.out_prefix}{fnum+1}.jpg", frame)
49         fnum += 1
50     else:
51         print(f"Reached {fnum} frames")
52         break
53     print(f"Wrote {fnum} frames under {args.out_prefix}*.jpg")
54 finally:
55     # Cleanup
56     cap.release()
57     cv.destroyAllWindows()
58
59 # %%
```

Listing 2: vid\_to\_imgs.py

The above script (listing 2) gives the following output (along with a GUI window to show the images of the video)

```

Folder ./out is being created
Reached 10 frames
Wrote 10 frames under ./out/img*.jpg
```

### 2.1.2 Images to Video

**Problem** Given a folder with images, create a video where the FPS (frame rate) can be adjusted

**Experiments & Learning** Experiments performed are listed below

1. Getting images into the program: It was decided that the user will place the images, labelled in a sorted order (numerically), in a dedicated folder. For this demo, the folder name is `./seq`
2. All user inputs are given through `argparse`
3. FPS will be handled by the `VideoWriter` in the saved video file. However, the preview FPS has to be manually adjusted (using `waitKey` delay)

**Solution** Store images in a numerical order from `1.jpg` to `N.jpg` (where `N` is any number) in a dedicated folder, like `./seq`. Then run the following (for 5 FPS)

```
python .\imgs_to_vid.py -i "./seq" -f 5.0 -o "./out-5.avi"
```

The code is shown in listing 3. A snapshot of the generated video is shown in figure 3a. The following is an example for 10 FPS

```
python .\imgs_to_vid.py -i "./seq" -f 10.0 -o "./out-10.avi"
```

A snapshot of the video generated is shown in figure 3b. See figure 3 for more information.

```

1 # Create video from images in a given folder
2 """
3     Given images (numerically sorted 1 through N) in a folder, read
4     them and create a video (no audio, only video)
5 """
6
7 # %% Import everthing
8 import cv2 as cv
9 import os
10 import glob
11 import argparse
12
13 # %% Argparse parser
14 parser = argparse.ArgumentParser(
15     formatter_class=argparse.ArgumentDefaultsHelpFormatter)
16 parser.add_argument('-i', '--imgs-folder', default='./seq',
17     help="Path to folder in which images are stored (all jpg files)")
18 parser.add_argument('-o', '--out-file', default='./out.avi',
19     help="Output file (uses AVI, XVID fourcc)")
20 parser.add_argument('-f', '--vid-fps', default=10.0, type=float,
21     help="The desired FPS (as float) for the output file")
22
23 # %% Main entrypoint
24 if __name__ == "__main__":
25     # Parse arguments
26     args, unknown_args = parser.parse_known_args()
27
28     # Variables
29     imgs_path = f"{os.path.realpath(args.imgs_folder)}/*.jpg"
30     fps = args.vid_fps
31     out_file = args.out_file
32     # File names (sorted numerically)
33     img_fnames = sorted(glob.glob(imgs_path), key=len)
34     shape_w_h = cv.imread(img_fnames[0]).shape[-2::-1] # (W, H) of video
35     # Video writer
36     fourcc = cv.VideoWriter_fourcc(*"XVID")
37     out_fhdlr = cv.VideoWriter(out_file, fourcc, fps, shape_w_h)
38
39     # For ever image found, write it to the video file
40     for i, img_file in enumerate(img_fnames):
41         # Read file
42         frame = cv.imread(img_file)
43         if frame is None:
44             print(f"Unable to read '{img_file}', skipping it!")
45             continue
46         # Write the image to video writer
47         out_fhdlr.write(frame)
48         # Preview
49         cv.imshow("Video", frame)
50         if cv.waitKey(int(1000/fps)) == ord('q'):
51             print(f"Quit received after {i} frames")
52             break
53     # Cleanup
54     out_fhdlr.release()
55     cv.destroyAllWindows()
56
57 # %%

```

Listing 3: imgs\_to\_vid.py

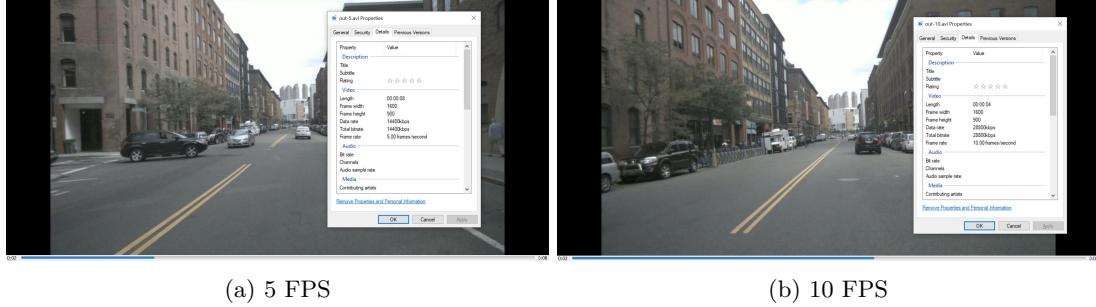
## 2.2 Capture Images through Webcam

**Problem** Use a webcam, capture images and save them to a folder

**Experiments & Learning** Experiments performed are listed below

1. Learned to capture images from a webcam using `VideoCapture`

**Solution** Run the following command



(a) 5 FPS

(b) 10 FPS

Figure 3: Saved videos

Image a is a screenshot of the 5 FPS playback (see the properties and the video frame). Image b is a screenshot of the 10 FPS playback. Notice that the 10 FPS version is faster, therefore a greater number of frames elapsed (for the 2 second point) compared to the 5 FPS version.

```
python .\webcam_capture.py -c 0 -o "./camimags"
```

The code is included in listing 4. Output is shown in figure 4.

```

1 # Capture images from a webcam
2 """
3     Given a webcam index and a destination folder (already existing),
4     preview the feed and capture images using it. Press 'c' to capture
5     an image and 'q' to quit.
6 """
7
8 # %% Import everything
9 import cv2 as cv
10 import os
11 import argparse
12
13 # %% Argument parser
14 parser = argparse.ArgumentParser(
15     formatter_class=argparse.ArgumentDefaultsHelpFormatter,
16     description="Press 'c' to capture, 'q' to quit")
17 parser.add_argument('-c', '--cam', default=0, type=int,
18     help="Camera index")
19 parser.add_argument('-o', '--out-folder', default="./camimags",
20     help="Output folder (should already exist)")
21
22 # %% Main entrypoint
23 if __name__ == "__main__":
24     args, extra_args = parser.parse_known_args()
25     webcam_id = args.cam
26     out_folder = args.out_folder
27
28     # Main camera work
29     cam = cv.VideoCapture(webcam_id)
30     imc = 0 # Image counter
31     while True:
32         ret, img = cam.read()
33         if not ret:
34             print("Camera did not give frames")
35             break
36         cv.imshow("Feed", img)
37         key = cv.waitKey(1)
38         if key == ord('q'):
39             print("Quit command received")
40             break
41         elif key == ord('c'):
42             imc += 1
43             fname = f"{out_folder}/{imc}.jpg"
44             print(f"Capturing image to '{fname}'")
45             cv.imwrite(fname, img)
46     cam.release()
47     cv.destroyAllWindows()
48
49 # %%

```

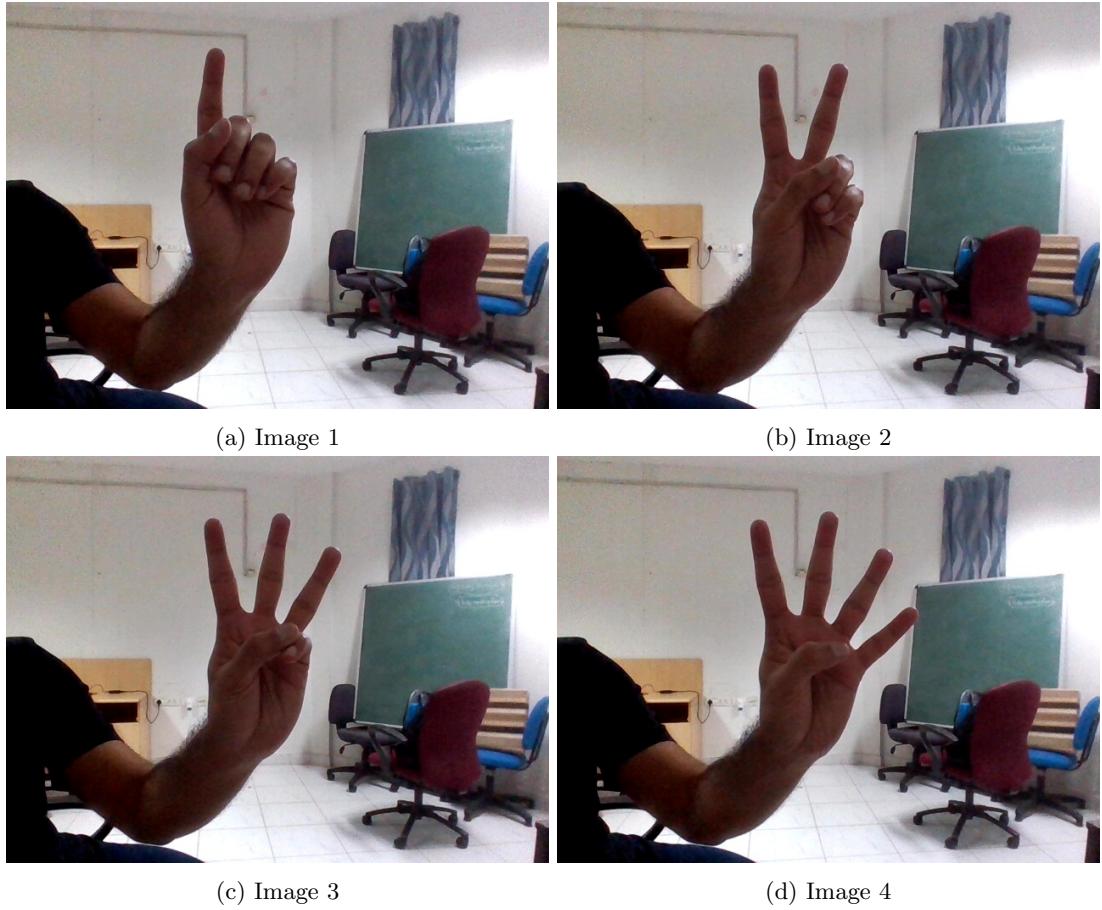


Figure 4: Images captured through webcam  
Four images captured through a webcam

Listing 4: `webcam_capture.py`

Four images were captured

```
Capturing image to './camimgs/1.jpg'
Capturing image to './camimgs/2.jpg'
Capturing image to './camimgs/3.jpg'
Capturing image to './camimgs/4.jpg'
Quit command received
```

### 2.3 Chroma Keying

**Problem Chroma Keying:** Given two videos (with one containing a green screen), create a fusion between the two, such that the subject of the green screen video is in the foreground and the green screen is replaced by the other video.

**Experiments & Learning** Experiments performed and things learned are listed below

1. Finding the video files was challenging. The desire was to have a foreground file where the green screen was consistent and easily extractable. Finally, a video was found on YouTube <sup>2</sup> and clipped. The video that would become the new background was also found on YouTube <sup>3</sup>. However, in the process of searching, datasets like VideoMatte240K were discovered (these weren't used because their size is too big for a single task).

<sup>2</sup><https://youtu.be/cqZuhJZAK-8>

<sup>3</sup><https://youtu.be/MPUBSZYESgU>

2. Comparisons were initially done *directly* on the green color channel (simple thresholding on green channel). This worked for the video, but is clearly not generalizable. There is no fixed metric that describes the *green-ness* of a color on the RGB color spectrum. Through some search and tutorials<sup>4</sup>, a system that uses the **HSV** (Hue, Saturation, Value) colorspace was implemented. *Hue* covers the color in a single number, *saturation* covers the sharpness (presence) of the color, and *value* covers the darkness of the color.
3. Through experiments, a suitable hue threshold and margin was selected. However, the program gives provision to use custom values also. Additionally, there is also provision to capture specific images (along with masks) *while* the program is running and previewing the result.

**Solution** The following script (in listing 5) can be run

```
python .\chroma_keying.py -g 215 -t 10 -r 25
```

The code can be found in listing 5 and frames from the resulting video are shown in figure 5

```

1 # Apply background based chroma keying
2 """
3     Given a foreground video with a greenscreen in background, a video
4     that should be underlaid as new background, and a green value (
5     along with hue threshold). Use '-h' option to know more.
6
7     More information at [1]
8
9     [1]: https://docs.opencv.org/4.x/df/d9d/tutorial\_py\_colorspaces.html
10 """
11
12 # %% Import everything
13 import cv2 as cv
14 import numpy as np
15 import argparse
16 import os
17
18 # %% Argparse variables
19 parser = argparse.ArgumentParser(
20     formatter_class=argparse.ArgumentDefaultsHelpFormatter,
21     description="Use 'q' to quit and 'c' to capture")
22 parser.add_argument('-g', '--g-val', default=216,
23     help="Green value (0-255)")
24 parser.add_argument('-t', '--hsv-thresh', default=5, type=float,
25     help="Threshold percentage for hue (0-100)")
26 parser.add_argument('-w', '--img-fg', default="./videos/tiger-gs.mp4",
27     help="Path to the foreground image")
28 parser.add_argument('-b', '--img-bg',
29     default="./videos/waterfall.mp4",
30     help="Path to the background image")
31 parser.add_argument('-o', '--out-folder', default="./chroma_out",
32     help="Folder name (to store results). Must be exiting.")
33 parser.add_argument('-r', '--out-fps', default=30,
34     help="FPS (writing)")
35
36 # %% Main entrypoint
37 if __name__ == "__main__":
38     args, extra_args = parser.parse_known_args()
39     g_val = args.g_val # Green value
40     h_thresh = args.hsv_thresh # Threshold (percentage)
41     # Foreground and background files
42     fg_file = os.path.realpath(args.img_fg)
43     bg_file = os.path.realpath(args.img_bg)
44     # Output folder and FPS
45     vid_out_file = os.path.realpath(args.out_folder)
46     out_fps = int(args.out_fps) # Output FPS
47     # Convert green value to HSV
48     h, s, v = cv.cvtColor(np.array([[0, g_val, 0]]], dtype=np.uint8),
49         cv.COLOR_BGR2HSV)[0,0] # Green -> HSV
50     print(f"HSV: {h}, {s}, {v}")
51
52     # Read a frame from both streams

```

---

<sup>4</sup>[https://docs.opencv.org/4.x/df/d9d/tutorial\\_py\\_colorspaces.html](https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html)

```

53     v1 = cv.VideoCapture(fg_file)
54     ret, img_fg = v1.read()
55     v1.release()
56     # Main task
57     i = 0
58     v1 = cv.VideoCapture(fg_file)
59     v2 = cv.VideoCapture(bg_file)
60     fourcc = cv.VideoWriter_fourcc(*"XVID")
61     vwrite = cv.VideoWriter(f"{{vid_out_file}}/chroma_res.avi", fourcc,
62         out_fps, img_fg.shape[-2::-1])
63     while True:
64         ret, img_fg = v1.read()
65         if not ret:
66             print("Foreground video finished")
67             break
68         ret, img_bg = v2.read()
69         if not ret:
70             print("Background video finished")
71             break
72     # Mask through HSV
73     img_fg_hsv = cv.cvtColor(img_fg, cv.COLOR_BGR2HSV)
74     lower_green = np.array([h*(1-h_thresh/100), 180, 180])
75     upper_green = np.array([h*(1+h_thresh/100), 255, 255])
76     # img_fg_hsv channels in range (AND of each range check)
77     mask = cv.inRange(img_fg_hsv, lower_green, upper_green)
78     # Construct final image
79     img_final = img_fg.copy()
80     img_final[mask == 255] = img_bg[mask == 255]
81     # Show result
82     cv.imshow("Fusion", img_final)
83     vwrite.write(img_final)
84     key = cv.waitKey(int(1000/30))
85     if key == ord('q'):
86         print("Exit signal received")
87         break
88     elif key == ord('c'):
89         i += 1 # New image
90         cv.imwrite(f"{{vid_out_file}}/mask{i}.jpg", mask)
91         cv.imwrite(f"{{vid_out_file}}/img_fg{i}.jpg", img_fg)
92         cv.imwrite(f"{{vid_out_file}}/img_bg{i}.jpg", img_bg)
93         cv.imwrite(f"{{vid_out_file}}/img_f{i}.jpg", img_final)
94         print(f"Saved image '{i}' in folder '{vid_out_file}'")
95     v1.release()
96     v2.release()
97     vwrite.release()
98     cv.destroyAllWindows()
99     print("Video writing completed")
100
101 # %%

```

Listing 5: chroma\_keying.py

The above code (listing) generates the following output

```

HSV: 60, 255, 215
Saved image '1' in folder '****\chroma_out'
Saved image '2' in folder '****\chroma_out'
Saved image '3' in folder '****\chroma_out'
Saved image '4' in folder '****\chroma_out'
Foreground video finished
Video writing completed

```

The output files are `chroma_res.avi` which contains the main video, `img_bgN.jpg` which contains the background image (which will be separated), `img_fg1.jpg` which contains the foreground image, `maskN.jpg` which contains the mask image, and `img_fN.jpg` which contains the final image. These are all summarized in figure 5.

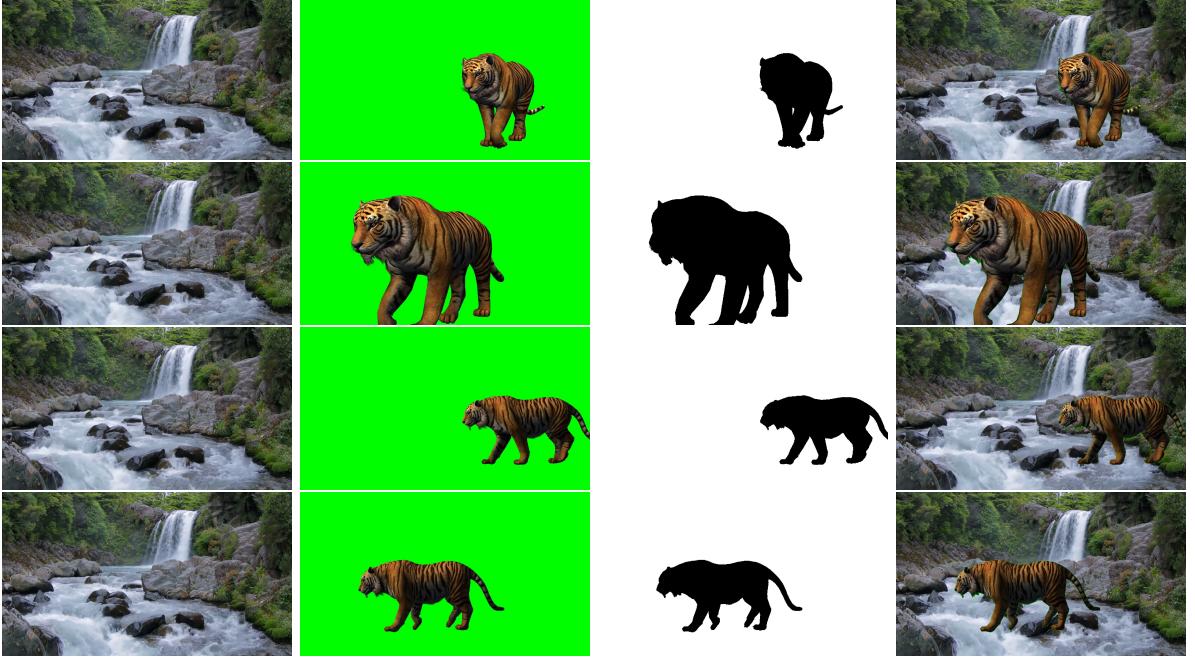


Figure 5: Chroma keying results

Snapshots and stages of the chroma keying process. The individual rows are the sequences (as time progresses). First column is the background that is to be substituted for the placeholder color. The second column is the foreground (with a placeholder color). The third column is the mask that detects the placeholder color (for substitution). The last column is the resulting image (which is put in a video file).

The video whose images are shown in the first column was taken from YouTube. The second column is also from YouTube.

### 3 Additional Tasks

This section is dedicated to discussion on *additional* tasks given in the assignment. This section can be skipped.

#### 3.1 Shot Detection

If the exact timings (frame numbers) of the shot transitions are given / known, then the problem is as simple as traversing through a list of indices. Only minor modifications to code in 2 will be required. This can be provided as a function.

If the timings (frame numbers) of the shot transitions are not known, then any of the following methods can be used

- **Color histogram:** Where color histogram differences are monitored and whenever they exceed a certain threshold, a scene change is detected. Some resources are [CNP06] and [MF03].
- **AI:** There are CNN based methods for this purpose. A reference is [XSX16].

The above methods yield the detection boundaries, which can be used in the function described in the beginning of this section.

#### 3.2 Face Detection

**Face Detection methods** The following are linked as references

- OpenCV facial detection using Haar Cascades.
- Some AI algorithms for facial detection can be found in [AWY17] and [ZZZ15].

The Face Detection Data Set can be found on <https://vis-www.cs.umass.edu/fddb/>, and downloaded through the procedure described in the GitHub repository cezs/FDDB.

## References

- [MF03] Jordi Mas and Gabriel Fernandez. “Video Shot Boundary Detection Based on Color Histogram.” In: *TRECVID*. 2003.
- [CNP06] Costas Cotsaces, Nikos Nikolaidis, and Ioannis Pitas. “Video shot detection and condensed representation. a review”. In: *IEEE signal processing magazine* 23.2 (2006), pp. 28–37.
- [ZZZ15] Stefanos Zafeiriou, Cha Zhang, and Zhengyou Zhang. “A survey on face detection in the wild: past, present and future”. In: *Computer Vision and Image Understanding* 138 (2015), pp. 1–24.
- [XSX16] Jingwei Xu, Li Song, and Rong Xie. “Shot boundary detection using convolutional neural networks”. In: *2016 Visual Communications and Image Processing (VCIP)*. 2016, pp. 1–4. doi: [10.1109/VCIP.2016.7805554](https://doi.org/10.1109/VCIP.2016.7805554).
- [AWY17] Yicheng An, Jiafu Wu, and Chang Yue. *CNNs for face detection and recognition*. 2017.