DIP Assignment 0 Report

Arghya Roy
2021115008

Libraries used

- `numpy`
- `cv2`
- `matplotlib`
- `collections`

Reading an image file into an array

I used the `imread` function of the `cv2` library to read the image files into `numpy` arrays.

Writing an array into an image file

I used the `imwrite` function of the `cv2` library to write the `numpy` arrays into image files at the designated paths.

Changing brightness of the image

Logic: Add `k` (hyperparameter) to each value in the array but while ensuring it stays in range i.e within the integers $0 - 255$.

$$I'(x, y) = k \cdot I(x, y)$$

Samples

Original images

Increased brightness

Decreased brightness

Changing contrast of the image

Logic:

- Firstly, I went for the simple approach of stretching the intensity range by making the maximum and minimum pixel values the upper and lower bounds respectively.

$$f_{ac}(a) = (a - a_{low}) \cdot \frac{255}{a_{high} - a_{low}}$$

- Then, I implemented contrast stretching to increase the contrast in the images. I took the top and bottom `p` (hyperparameter) percentile pixel values and stretched the intensity range to make these two values the upper and lower bounds. This gave me better results.

Samples

Original images

Contrast Stretched images

Changing a color image to grayscale

I tried two approaches for changing a color image to grayscale on the following color image.

Approaches:

1. $\text{Grayscale} = \alpha \times R + \beta \times G + \gamma \times B$
2. $\text{Grayscale} = \frac{R+G+B}{3}$

Results for 1 and 2 respectively

Observations

Both the grayscale images seem very similar. However, it seems like for the second approach (averaging the value of the three channels), the background seems slightly more evenly darker.

Converting a grayscale image to color using pseudo mapping

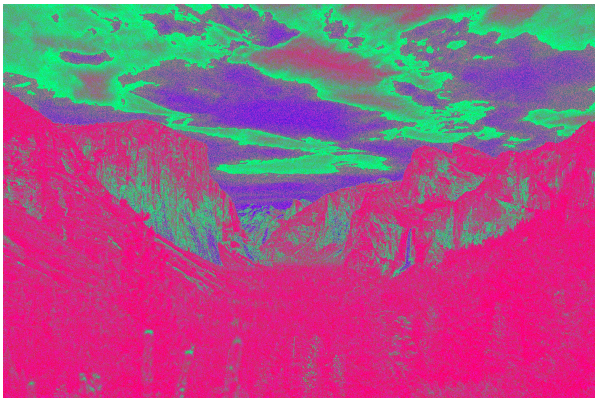
Logic

- I came up with a random color map
  - $\text{if pixel intensity} < 128:$

    $\rightarrow [255 \times rand(0,1), 2 \times \text{pixel intensity}, 255 - 2 \times \text{pixel intensity}]$
  - else

    $\rightarrow [\text{pixel intensity}, 255 - \text{pixel intensity}, 255 \times rand(0,1)]$
- I applied the color map on all pixels while ensuring they're in range.

Result



Replacing green screen with different background

Logic

- I detected the color of the green screen by finding the most frequently occurring color.
- The central idea was to place the other pixels of the foreground as the corresponding pixels of the background.
- However, because of factors like shadows, etc., it will not always be that *exact* color; so, I added a value `tolerance` (hyperparameter) for the range to be more flexible, but while ensuring the values stay within range.

Results

Inputs (foreground and background respectively)

Output

Observations

While it seems to have done a decent job, slight bits of the green screen are still visible. While it may seem that simply increasing the `tolerance` can make it better, it starts tampering the tree since it is green in color, though a darker shade.

Reading a video file into an array of images and writing it back as a video

I used the `VideoCapture` function of the `cv2` library to read the image and then stored the frames in a list, following which I used the `VideoWriter` function to write the frames into the video, which I configured as `mp4` by using the `VideoWriter_fourcc` function.

Creating a 1 second transition video from an image to another

Logic:

- I essentially just used the formula

$$F_t(x,y) = (1-t)\cdot I_1(x,y) + t\cdot I_2(x,y)$$
$$t = \frac{n}{N}; n = 0...N$$

- I also used the same library functions like the previous task to create the video.

Result

The transition video is available here: https://github.com/CS7-404-DIP-M24/A0-arghyawning/tree/main/media/outputs/transition

---