Assignment1 / {Filters}

Graphics Programming
Tristan Goodell

Input Images

image1.png

“Bridal Veil Falls”
750x1000

Greyscale / {Pic_2_1}

- *Task:* Make image1 greyscale with 20/70/10 model.

image1.png

pic_2_1.png

Greyscale / {Pic_2_1} / [code]

```
1  def greyscale(img):
2      b = img[:, :,0]*0.1
3      g = img[:, :,1]*0.7
4      r = img[:, :,2]*0.2
5      img=b+g+r
6      return img
```

blackWhite / {Pic_2_2}

- *Task:* Make image1 black & white w/ threshold=128.

image1.png

pic_2_2.png

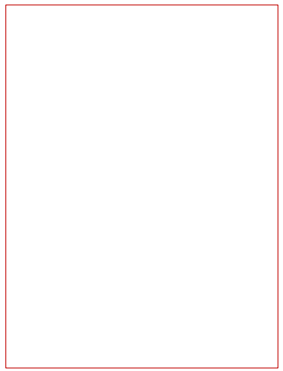
blackWhite / {Pic_2_2} / [code]

```
1   def blackWhite(img, threshold):
2       bw = 1*img[:,:,1]
3       bw[np.uint8(bw) < threshold] = 0
4       bw[np.uint8(bw) > threshold] = 255
5       return bw
```

blackWhite / {Pic_2_2_n}

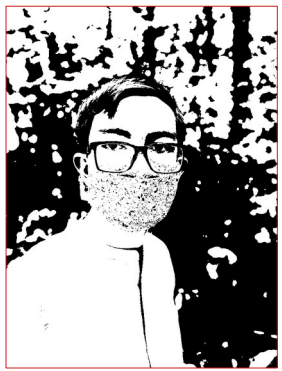
- *Task:* Use threshold -1 to 255 with increments of 32.


image1.png


pic_2_2_0.png


pic_2_2_1.png


pic_2_2_2.png


pic_2_2_3.png

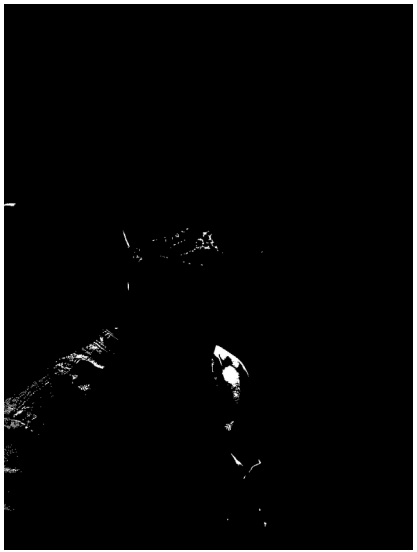

pic_2_2_4.png


pic_2_2_5.png

blackWhite / {Pic_2_2_n} / Cont.

- *Task:* Use threshold -1 to 255 with increments of 32.

pic_2_2_6.png



pic_2_2_7.png

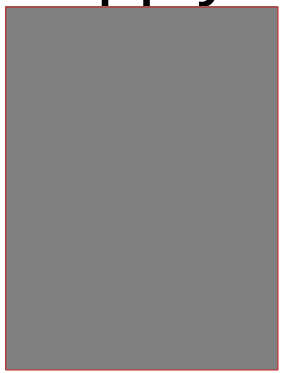
blackWhite / {Pic_2_2_n} / [code]

```
1   # 2.2b Apply blackWhite filter using threshold values from -1 to 255 in increments of 32.
2   for i in range(1, 9):
3       pic_2_2b=blackWhite(img, (32*i)-1)
4       cv2.imwrite("output/pic_2_2_" + str(i) + ".png", pic_2_2b)
```

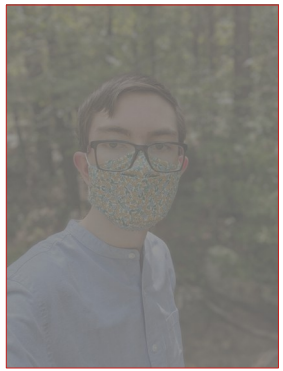
Desaturate / {Pic_2_3_n}

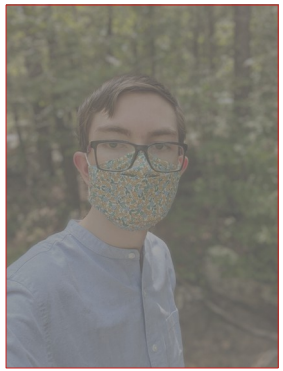
- *Task:* Apply the desaturate filter 0 to 1, 0.1 increments.

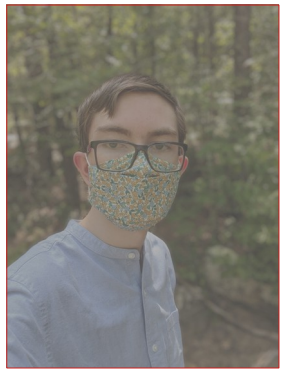

image1.png


pic_2_3_0.png


pic_2_3_1.png


pic_2_3_2.png


pic_2_3_3.png


pic_2_3_4.png


pic_2_3_5.png

Desaturate / {Pic_2_3_n} / Cont.

- *Task:* Apply the desaturate filter 0 to 1, 0.1 increments.


image1.png


pic_2_3_6.png


pic_2_3_7.png


pic_2_3_8.png


pic_2_3_9.png


pic_2_3_10.png

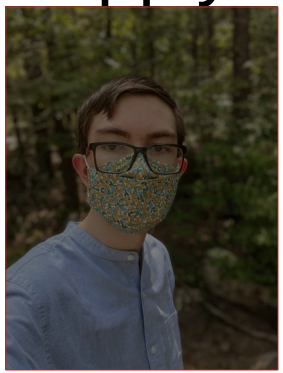
Desaturate / {Pic_2_3_n} / [code]

```
1   def desaturate(img ,percent):
2       # Set contra to a double and img for overflow reasons
3       desat = np.double(img[:, :, :])
4
5       # Actual math behind desat
6       desat[:, :, :] = 1.0 * (desat[:, :, :] - 128) * percent + 128
7
8       # Overflow Check
9       desat[desat > 255] = 255
10      desat[desat < 0] = 0
11
12      return np.uint8(desat)
```

Contrast / {Pic_2_4_n}

- *Task:* Apply the contrast filter 0.5 to 1.5, 0.1 increment.


image1.png


pic_2_4_0.png


pic_2_4_1.png


pic_2_4_2.png


pic_2_4_3.png


pic_2_4_4.png


pic_2_4_5.png

Contrast / {Pic_2_4_n} / Cont.

- *Task:* Apply the contrast filter 0.5 to 1.5, 0.1 increment.


image1.png


pic_2_4_6.png


pic_2_4_7.png


pic_2_4_8.png


pic_2_4_9.png


pic_2_4_10.png

Contrast / {Pic_2_4_n} / [code]

```
1   def contrast(img, factor):
2       cimg = 1*img
3       b = cimg[:, :, 0]
4       g = cimg[:, :, 1]
5       r = cimg[:, :, 2]
6
7       b = b * factor
8       b[b > 255] = 255
9       b[b < 0] = 0
```

```
1   g = g * factor
2   g[g > 255] = 255
3   g[g < 0] = 0
4
5   r = r * factor
6   r[r > 255] = 255
7   r[r < 0] = 0
8
9   cimg[:, :, 0] = b
10  cimg[:, :, 1] = g
11  cimg[:, :, 2] = r
12
13  return cimg
```

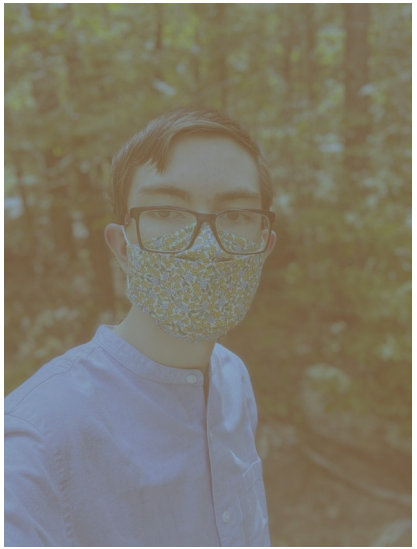
Tint / {Pic_2_5_n}

- *Task:* Create three tinted photos.

image1.png
original

pic_2_5_0.png
50% Blue

pic_2_5_1.png
70% Green



pic_2_5_2.png
90% Red

Tint / {Pic_2_5_n} / [code]

```
def tint(img, color, percent):
    timg=img[:,:,:]
    tint=127

    # Assiging bgr values
    b=timg[:, :, 0]
    g=timg[:, :, 1]
    r=timg[:, :, 2]

    # Applying tints based on color input & percent.
    if color=="blue":
        b = (1 - percent) * b + percent * tint
    if color=="green":
        g = (1 - percent) * g + percent * tint

    if color=="red":
        r = (1 - percent) * r + percent * tint

    # Reassigning gbr values
    timg[:, :, 0] = b
    timg[:, :, 1] = g
    timg[:, :, 2] = r

    return timgt
```