Assignment6 / {WebCamBooth}

Graphics Programming / Tristan Goodell

Filter 1

Filter 1 - {code}

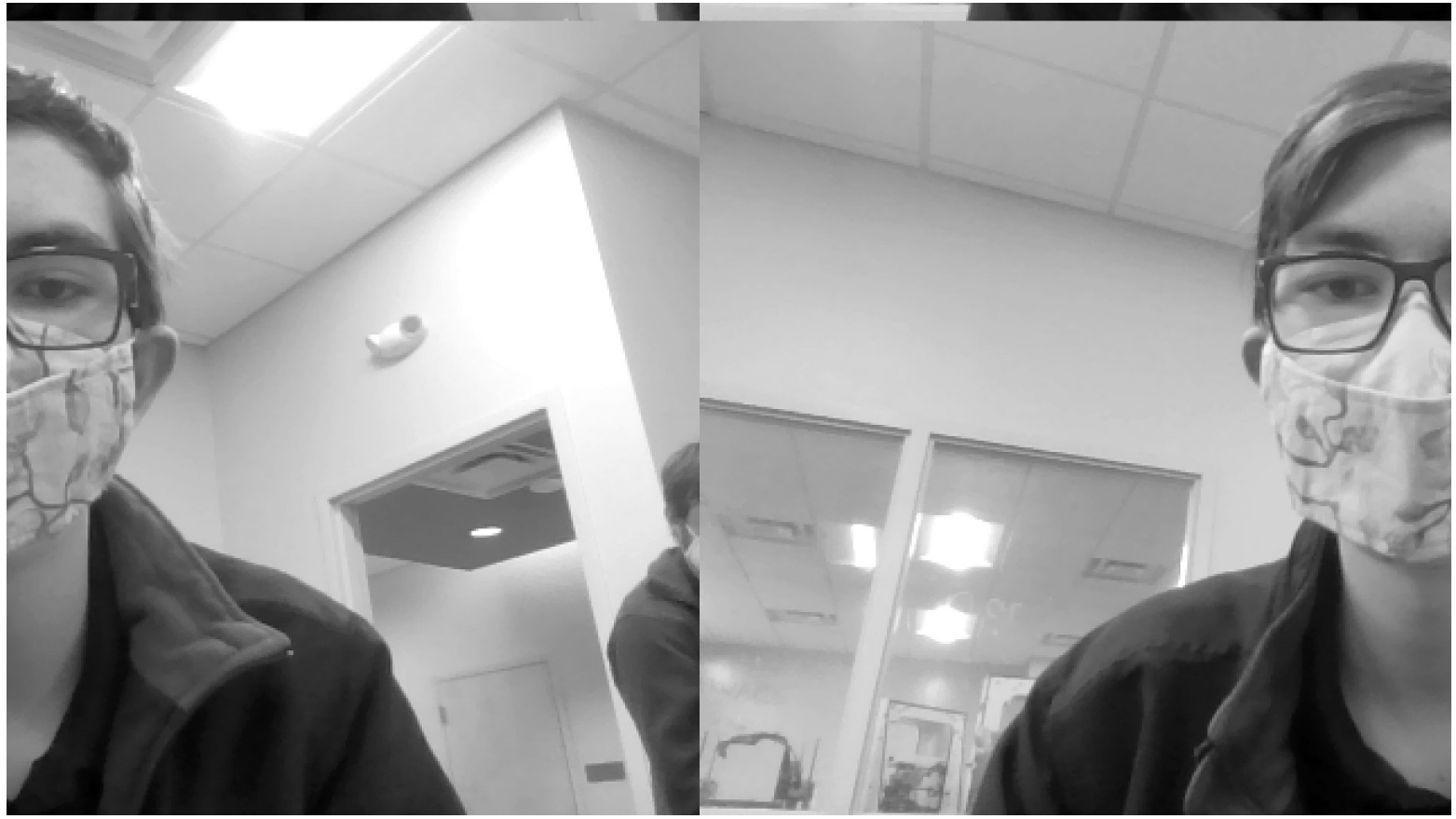
```
def desaturate(img):
    desat = 1*np.double(img[:, :, :])
    greyImg = greyscale(img)

    # Actual math behind desat
    desat[:, :, :] = (desat[:, :, :] *(1 - 0.5)) + (greyImg[:,:,None] * 0.5)
    # Overflow Check
    desat[desat > 255] = 255
    desat[desat < 0] = 0

    return desat
```

Filter 2

Filter 2 - {code}

```
if warp==True:
    angle = np.pi * t / 1000 + np.pi / 3 * np.cos(t * .05)*2
    T1 = np.float64([[1, 0, w/3*np.cos(angle)],
                     [0, 1, h/3*np.cos(angle)],
                     [0, 0, 1]])
    frame = cv2.warpPerspective(frame, T1, (w, h))
```

Filter 3



```
if nonlinear==True:
    frame=cv2.remap(frame,(x+dx*t)%(w-1),(y+dy*np.sin(np.pi*t/180))%(h-1),cv2.INTER_CUBIC)
    frame=normalize(greyscale(frame))
```

Filter 4

Filter 4 - {code}

```
if matrix==True:
    M = np.float32([[1, 0, np.sin(2*t+10)],
                    [0, 1, np.sin(2*t+10)],
                    [0, 0, 1]])
    frame = cv2.warpPerspective(frame, M, (w,h))
    frame = cv2.remap(frame, x + dx * np.sin(np.pi * (y + t)/180),y + dy * np.sin(np.pi * (x + t)/180), 0)
```

Filter 5

Filter 5 - {code}

```
if eyez==True:
    ret, frame = cap.read()
    eyes = eyecacade.detectMultiScale(frame[:, :, 1], scaleFactor=1.2, minNeighbors=5)
    # cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),5)

    yellow = (0, 255, 255)
    for (x, y, w, h) in eyes:
        star(frame,w//2,(x+w//2,y+h//2),color=yellow)

    c=0
    while c<3:
        star(frame, random.randint(1,5), (random.randint(0,640), random.randint(0,360)), color=yellow)
        c+=1
```

Filter 6

Filter 5 - {code}

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