HW1

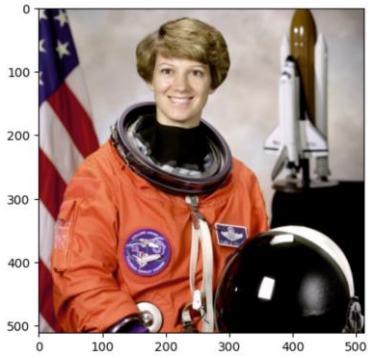
2022019734 김민서

1. Projective Image Transformation

1-1. Astronaut image

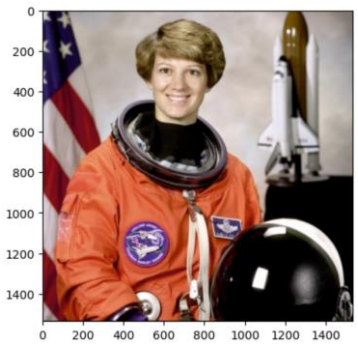
1) Original

skimage.data.astronaut()로 이미지를 불러오면, 아래와 같이 표시되는 것을 알 수 있다.



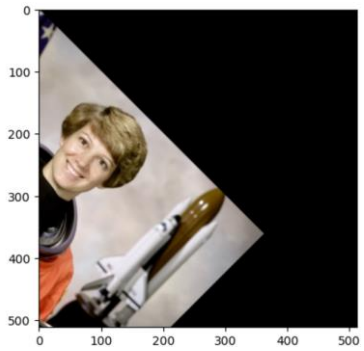
2) Scaling

Transformation Matrix로 $T = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$를 사용한 뒤, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 3배만큼 커진 것을 알 수 있다.



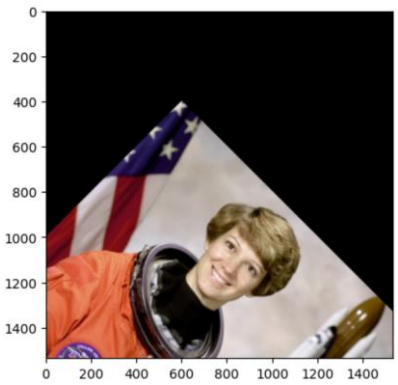
3) Rotation

Transformation Matrix로 $T = \begin{pmatrix} \cos\left(\frac{\pi}{4}\right) & -\sin\left(\frac{\pi}{4}\right) & 0 \\ \sin\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) & 0 \\ 0 & 0 & 1 \end{pmatrix}$를 쓰고, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 $\frac{\pi}{4}$만큼 회전한 것을 알 수 있다.



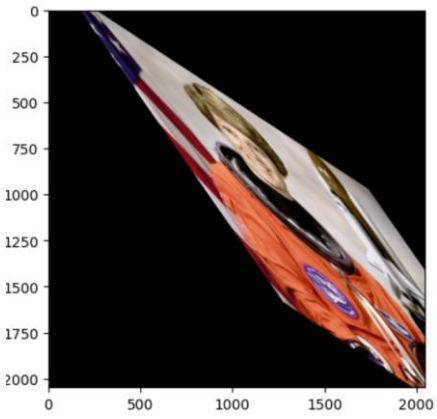
4) Similarity transform

Transformation Matrix로 $T = \begin{pmatrix} 3\cos\left(\frac{\pi}{4}\right) & -3\sin\left(\frac{\pi}{4}\right) & 600 \\ 3\sin\left(\frac{\pi}{4}\right) & 3\cos\left(\frac{\pi}{4}\right) & 400 \\ 0 & 0 & 1 \end{pmatrix}$를 사용한 후, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 3배만큼 커지고, $\frac{\pi}{4}$만큼 회전했으며, x축 방향으로 600, y축 방향으로 400만큼 이동한 것을 알 수 있다.



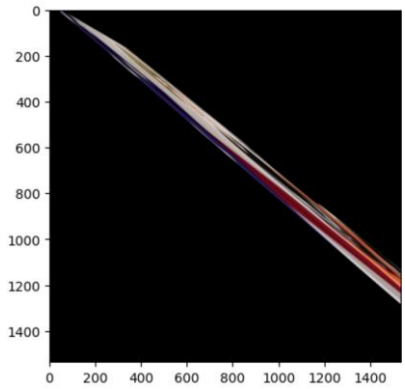
5) Affine transform

Transformation Matrix로 $T = \begin{pmatrix} 3.2 & 2.3 & 100 \\ 2.1 & 3.3 & -100 \\ 0 & 0 & 1 \end{pmatrix}$를 사용한 후, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 바뀐 것을 볼 수 있고, 이때, 이미지의 parallelism은 유지되는 것을 확인할 수 있다.



6) Projective transform

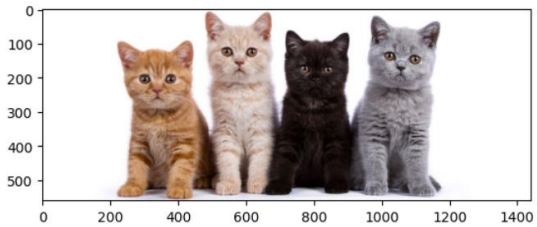
Transformation Matrix로 $T = \begin{pmatrix} 2.3 & 4.2 & 12 \\ 1.2 & 3.7 & -21 \\ 0.005 & -0.003 & 1 \end{pmatrix}$ 를 사용한 후, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 바뀐 것을 볼 수 있다.



1-2. My image

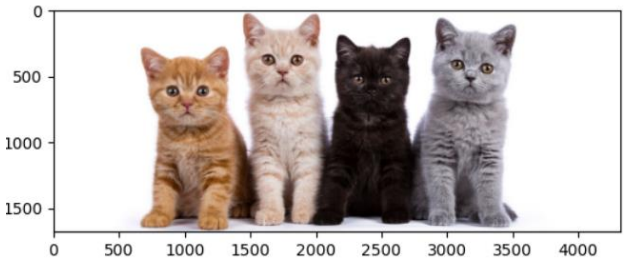
1) Original

cv2.imread()로 이미지를 읽어오고, cv2.cvtColor()로 format을 BGR에서 RGB로 바꿔주면 아래와 같은 고양이 사진을 확인할 수 있다.



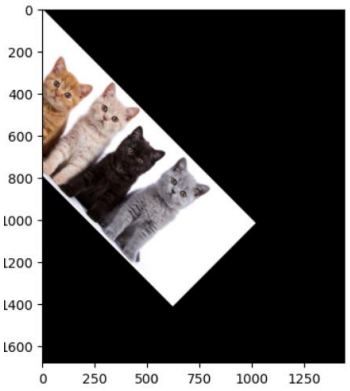
2) Scaling

Transformation Matrix로 $T = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$를 사용한 뒤, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 3배만큼 커진 것을 알 수 있다.



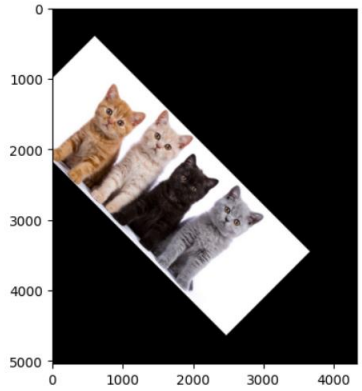
3) Rotation

Transformation Matrix로 $T = \begin{pmatrix} \cos\left(\frac{\pi}{4}\right) & -\sin\left(\frac{\pi}{4}\right) & 0 \\ \sin\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) & 0 \\ 0 & 0 & 1 \end{pmatrix}$를 쓰고, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 $\frac{\pi}{4}$만큼 회전한 것을 알 수 있다.



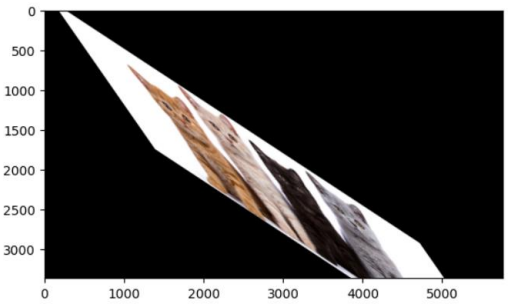
4) Similarity transform

Transformation Matrix로 $T = \begin{pmatrix} 3\cos\left(\frac{\pi}{4}\right) & -3\sin\left(\frac{\pi}{4}\right) & 600 \\ 3\sin\left(\frac{\pi}{4}\right) & 3\cos\left(\frac{\pi}{4}\right) & 400 \\ 0 & 0 & 1 \end{pmatrix}$를 사용한 후, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 3배만큼 커지고, $\frac{\pi}{4}$만큼 회전했으며, x축 방향으로 600, y축 방향으로 400만큼 이동한 것을 알 수 있다.



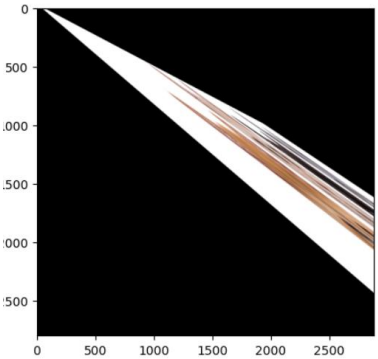
5) Affine transform

Transformation Matrix로 $T = \begin{pmatrix} 3.2 & 2.3 & 100 \\ 2.1 & 3.3 & -100 \\ 0 & 0 & 1 \end{pmatrix}$를 사용한 후, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 바뀐 것을 볼 수 있고, 이때, 이미지의 parallelism은 유지되는 것을 확인할 수 있다.



6) Projective transform

Transformation Matrix로 $T = \begin{pmatrix} 2.3 & 4.2 & 12 \\ 1.2 & 3.7 & -21 \\ 0.005 & -0.003 & 1 \end{pmatrix}$를 사용한 후, cv2.warpPerspective()를 사용하면, 아래와 같이 이미지가 바뀐 것을 볼 수 있다.

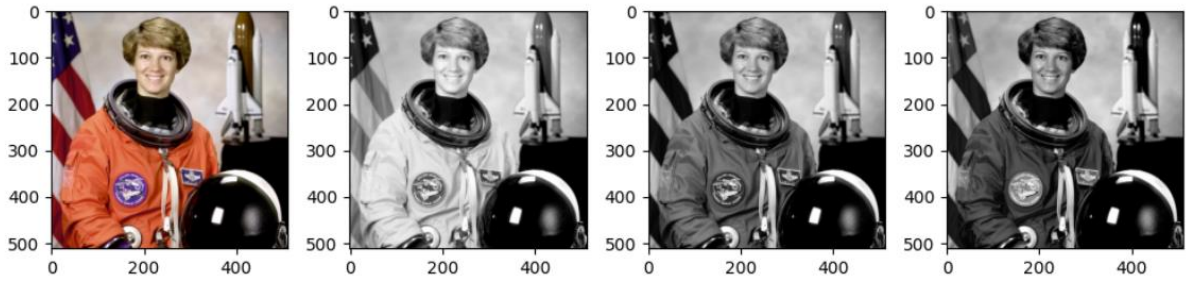


2. Color Space

1-1. Astronaut image

1) RGB

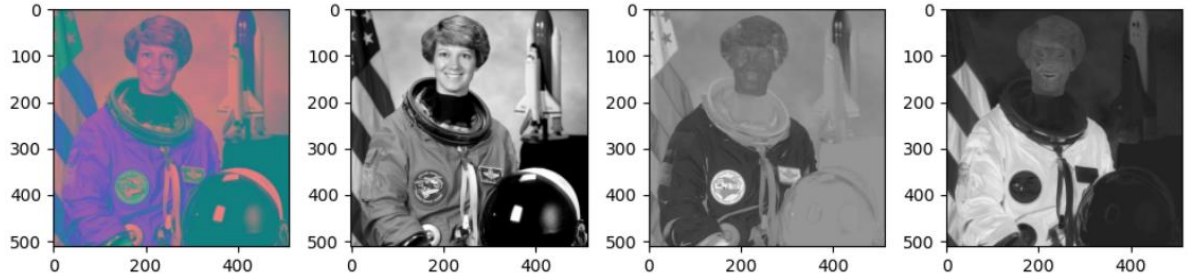
skimage.data.astronaut()로 이미지를 불러온 후, R, G, B = img[:, :, 0], img[:, :, 1], img[:, :, 2]로 rgb 채널을 분리한 후, 각각의 채널을 흑백으로 나타내면 아래와 같이 나타낼 수 있다.



2) RGB to YCbCr

$$
\begin{aligned}
Y' &= 16 + \frac{65.481 \cdot R'_D}{255} + \frac{128.553 \cdot G'_D}{255} + \frac{24.966 \cdot B'_D}{255} \\
C_B &= 128 - \frac{37.797 \cdot R'_D}{255} - \frac{74.203 \cdot G'_D}{255} + \frac{112.0 \cdot B'_D}{255} \\
C_R &= 128 + \frac{112.0 \cdot R'_D}{255} - \frac{93.786 \cdot G'_D}{255} - \frac{18.214 \cdot B'_D}{255}
\end{aligned}
$$

위의 수식에 따라서 RGB coordinate을 YCbCr로 변환한 뒤, 각각의 채널을 흑백으로 나타내면 아래와 같이 나타낼 수 있다.



3) RGB to HSI

$$H = \begin{cases} \theta & \text{if } B \le G \\ 2\pi - \theta & \text{if } B > G \end{cases}$$

Hue component formula

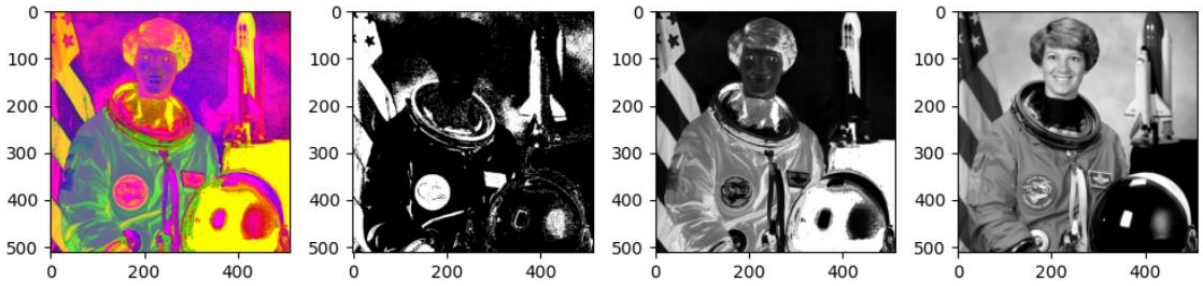
$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)]$$

Saturation component formula

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\}$$

$$I = \frac{1}{3}(R+G+B)$$

R, G, B 채널에 각각 255를 나눠서 normalize를 해준 뒤, 위 공식에 대입하면 HSI로 변환할 수 있고, 각각의 채널을 흑백으로 나타냈을 때 그 결과는 아래와 같다.



4) Compare Characteristics

RGB의 경우, intensity에 대한 정보가 따로 나뉘어져 있지 않고, rgb 채널에 나뉘어져 담겨있기 때문에, 각각의 채널을 흑백으로 나타내더라도 크게 이상하지 않고, 모든 채널에서 물체의 형태를 비교적 정확하게 파악할 수 있었다. 한편, YCbCr과 HSI는 intensity에 대한 정보가 각각 Y 채널과 I 채널에 담겨있기 때문에, 해당 채널들에서는 흑백으로 나타낸 사진이 크게 이상하지 않고, 물체의 형태도 비교적 정확하게 파악할 수 있으나, 이외의 채널에서는 흑백으로 나타낸 사진이 일반적인 흑백사진과는 동떨어져 있고, 물체의 형태를 정확하게 파악하기 힘든 경우가 있었다. YCbCr과 HSI를 비교해보면, HSI의 경우 H 채널만으로 물체의 색깔이 빨간색, 초록색, 파란색 중 어디에 가까운지 바로 알 수 있지만, YCbCr의 경우 Cb 채널과 Cr 채널을 같이 이용해야 해당 정보를 더욱 정확하게 얻을 수 있었다.

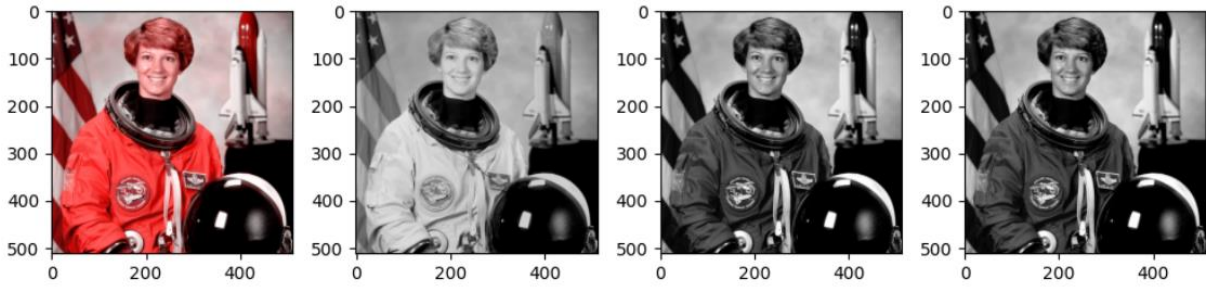
5) HSI to RGB

$$B = I(1-S) \qquad R = I(1-S) \qquad G = I(1-S)$$

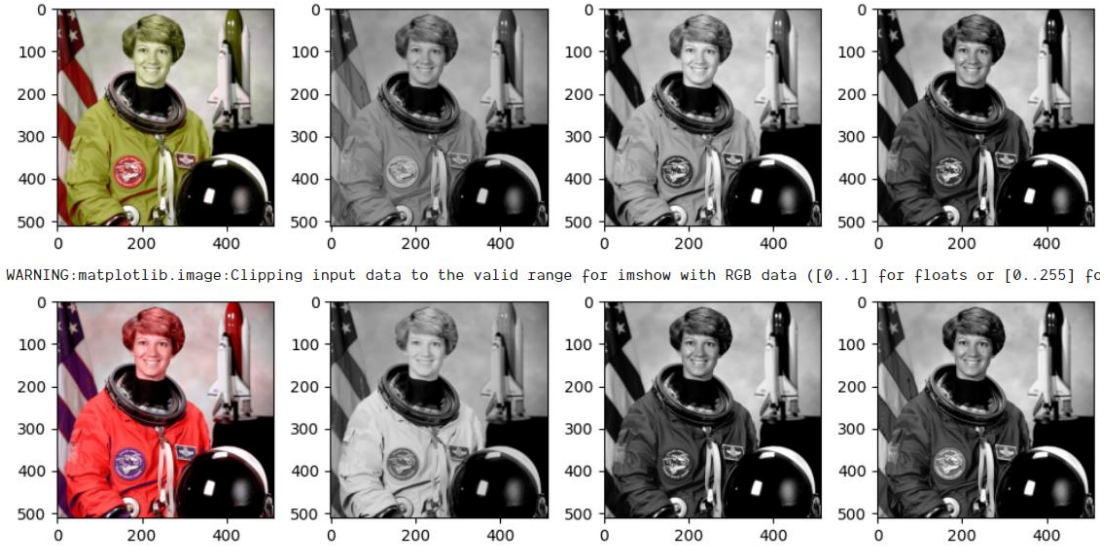
$$R = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right] \quad G = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right] \quad B = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right]$$

$$G = 3I - (R+B) \qquad B = 3I - (R+G) \qquad R = 3I - (G+B)$$

H 채널의 범위에 따라서 120도 이하일 경우에는 왼쪽의 공식을 적용하고, 120도와 240도 사이일 경우에는 H 값에서 120을 빼준 뒤 가운데 공식을 적용하며, 240도 이상일 경우에는 240을 빼준 뒤 오른쪽 공식을 적용했다. 이때, 복원된 결과는 아래의 이미지와 같다.

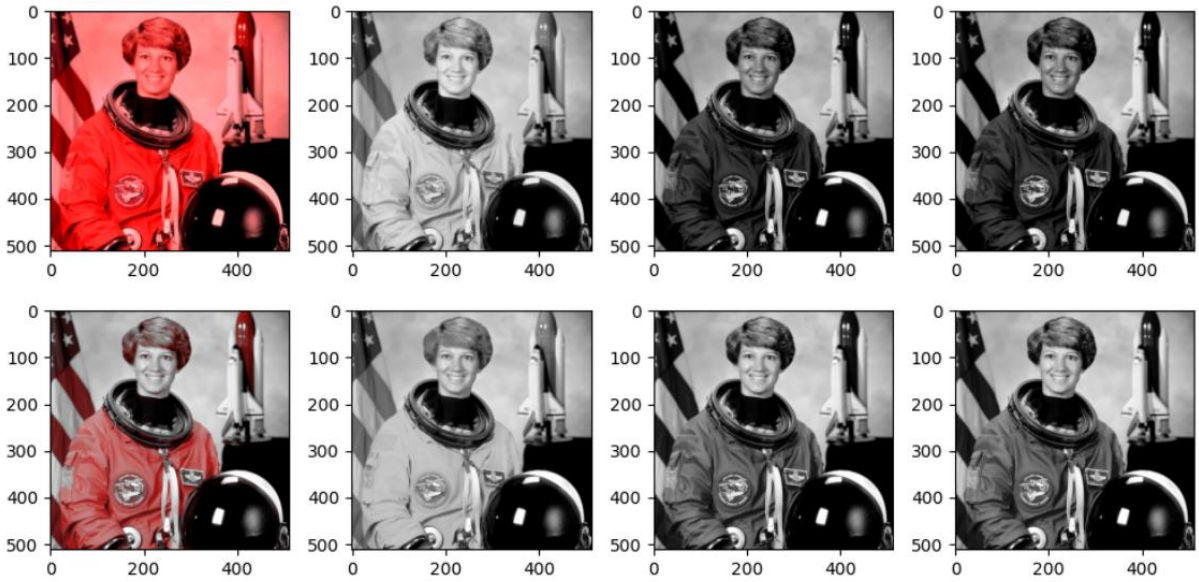


H 채널의 값을 전반적으로 내린 뒤 RGB로 변환했을 때는 비교적 초록색이 강조된 이미지를 얻은 반면, H 채널 값을 올렸을 때는 비교적 빨간색이 더 강조된 이미지를 얻을 수 있었다.

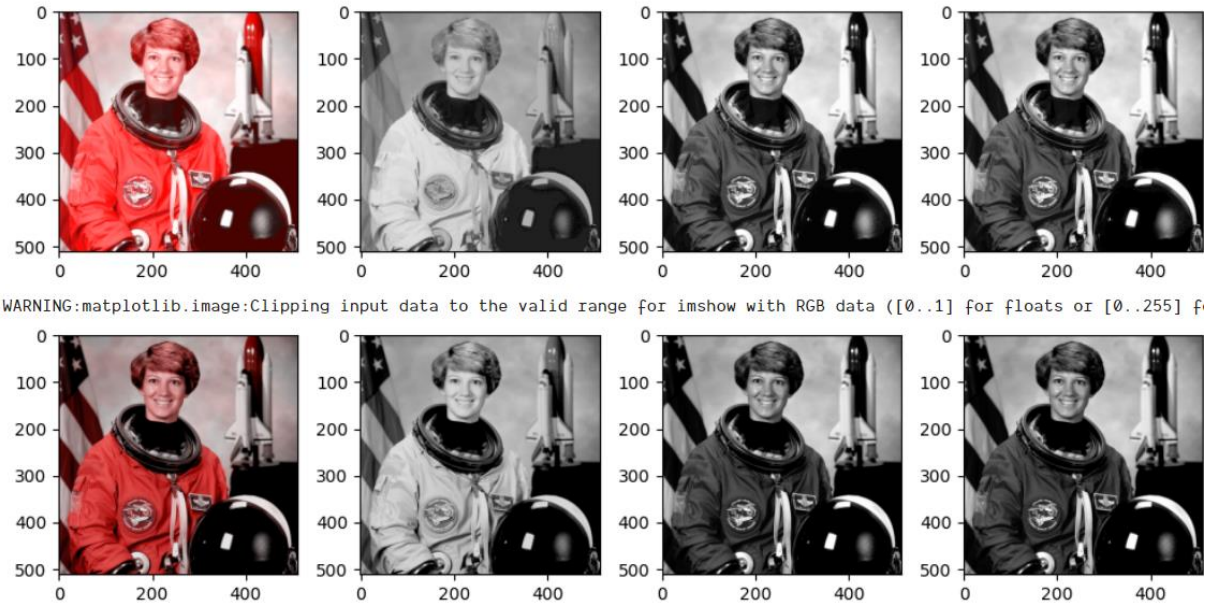


```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] fo
```

한편, S 채널의 값을 올렸을 때는 채도가 상당히 강한 이미지를 얻을 수 있었고, S 채널의 값을 내렸을 때는 전반적으로 채도가 낮은, 물 빠진 듯한 색감의 이미지를 얻었다.



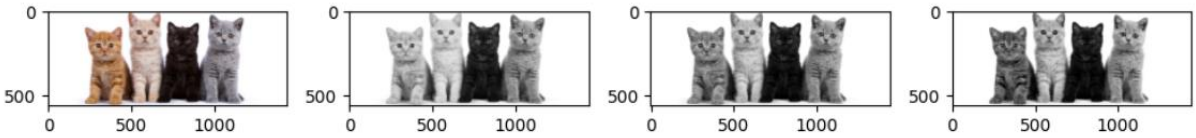
마지막으로, I 채널의 값을 올렸을 때는 전반적으로 밝은 이미지를 얻을 수 있었지만, I 채널의 값을 내렸을 때는 비교적 어두운 이미지를 얻을 수 있었다.



1-2. My image

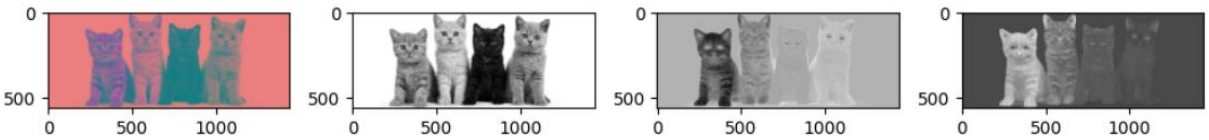
1) RGB

cv2.imread()로 이미지를 읽어오고, cv2.cvtColor()로 format을 BGR에서 RGB로 바꿔준 뒤 R, G, B = img[:, :, 0], img[:, :, 1], img[:, :, 2]로 rgb 채널을 분리한 후, 각각의 채널을 흑백으로 나타내면 아래와 같이 나타낼 수 있다.



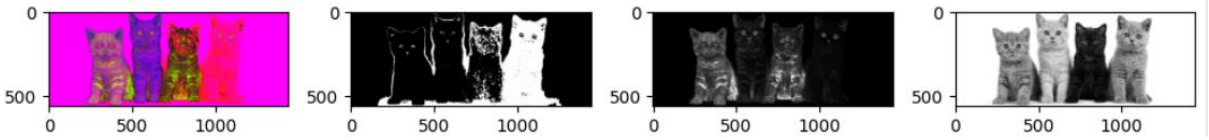
2) RGB to YCbCr

앞서 언급한 수식에 따라서 RGB coordinate을 YCbCr로 변환한 뒤, 각각의 채널을 흑백으로 나타내면 아래와 같이 나타낼 수 있다.



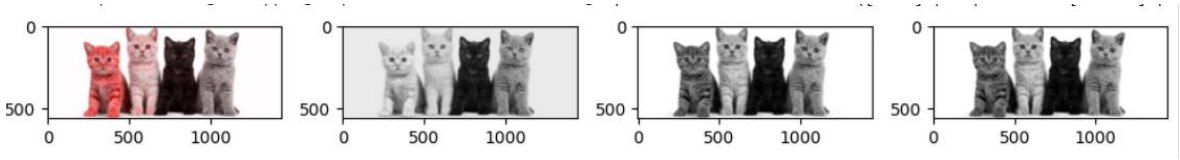
3) RGB to HSI

앞서 언급한 수식에 따라서 RGB coordinate을 HSI로 변환한 뒤, 각각의 채널을 흑백으로 나타내면 아래와 같이 나타낼 수 있다.

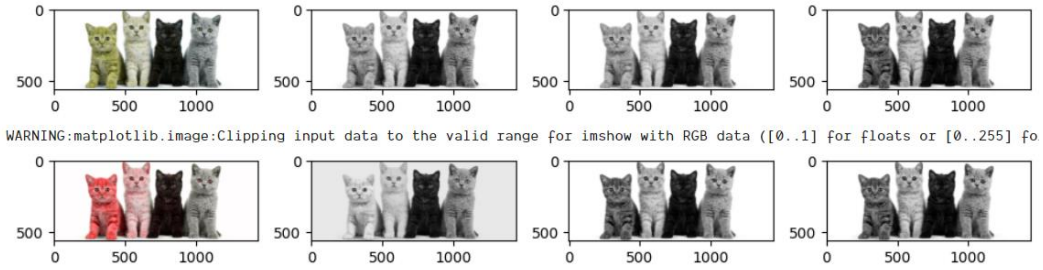


4) HSI to RGB

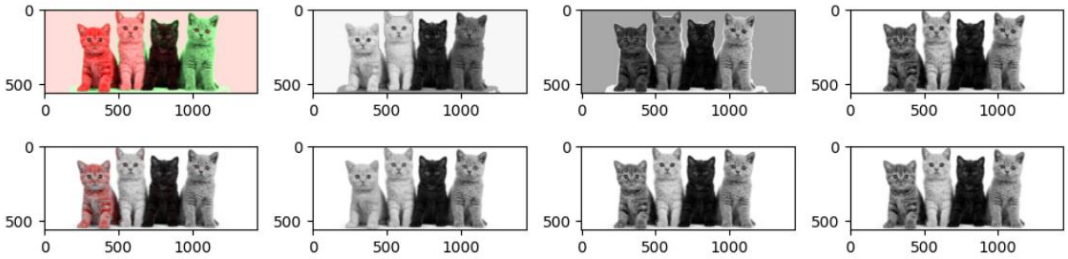
앞서 언급한 수식에 따라서 HSI coordinate을 RGB로 변환한 뒤, 각각의 채널을 흑백으로 나타내면 아래와 같이 나타낼 수 있다.



H 채널의 값을 전반적으로 올렸을 때는 초록색이 강조된 사진을, 내렸을 때는 붉은색이 강조된 사진을 얻을 수 있었다.



S 채널의 값을 전반적으로 올렸을 때는 채도가 굉장히 높은 사진을 얻었다. 특히, 제일 오른쪽 고양이의 경우, 기존에 있던 초록색 기운이 돌던 회색의 채도가 너무 높아진 나머지 형광 초록색으로 표현이 됐다. 반면 값을 내렸을 때는 채도가 낮은 물빠진 듯한 색감의 사진을 얻을 수 있었다.



마지막으로, I채널의 값을 올렸을 때는 전체적으로 밝은 사진을, 내렸을 때는 전체적으로 어두운 사진을 얻을 수 있었다.