

## [ 클릭 시 바뀌는 이미지의 구현 ]

인터넷을 보면, 클릭하게 되면 원본과는 다른 이미지가 보이는 png파일들이 있다. 이는 png파일 포맷은 투명도를 포함하여, 그 배경색이 달라지면 표출되는 픽셀의 색이 달라질 수 있음을 응용한 것이다. 이것을 직접 구현해보았다.

먼저, 이미지의 어떤 픽셀의 표출되는 RGB 값이  $x$ 이고, 이미지의 배경색이 바뀌었을 때 새롭게 바뀐 값이  $y$ 라고 하자. 이때  $x, y$ 는 3차원의 벡터가 된다. 이때 이  $x$ 는 png이미지의 표출되는 픽셀 값이므로, 실제 픽셀 값과 투명도의 곱으로 주어진다. 실제 픽셀 값을  $v$ , 투명도를  $a$ 라 하자.  $v$ 는 3차원의 벡터,  $a$ 는 스칼라이다. 이때 원래 배경색과 바뀐 배경색을 각각  $b_x, b_y$ 라 두자. 그렇다면 각 변수 사이에 아래의 관계가 성립한다.

$$x = (v - b_x)a + b_x$$

$$y = (v - b_y)a + b_y$$

직관적으로,  $a = 0$ 이면 배경색이 되며,  $a = 1$ 이면 실제 픽셀 값을 표시한다.

이때 배경색에 따라 이미지가 달라지는 그림을 만드는 것은 아래와 같은 문제이다.

$$x, y, b_x, b_y \text{ 가 주어질 때 } v, a \text{ 를 찾을 수 있는가?}$$

만약 모든 픽셀 값들이 스칼라라면 변수가 2개인 연립방정식이 되므로 아래와 같이 유일하게 해를 구할 수 있다. 기존의 이미지들은 모두 흑백 이미지였는데, 이런 이유 때문이라고 짐작된다.

$$\begin{aligned} a(v - b_x) &= x - b_x \\ a(v - b_y) &= y - b_y \\ ab_x - ab_y &= y - x + b_x - b_y \\ \therefore a(b_x - b_y) &= y - x + (b_x - b_y) \\ \therefore a &= \frac{y - x}{b_x - b_y} + 1 \end{aligned}$$

그러나  $x, y, b$  등이 벡터라면, 연립방정식이 6개 있는 것과 같으므로, 이를 만족시키는  $a$ 의 일반해가 존재하지 않는다. 따라서  $a(b_x - b_y) - y + x - b_x + b_y = 0$ 의 최대한 근사한 해를 찾아 대체하기로 하였다. 먼저 상수를 정리하여 식을 간단하게  $ax + b = 0$ 로 표현하였다. 여기에서  $a, b$ 는 위의  $a, b_x, b_y$ 와는 무관하다. 그렇다면 이는 곧 점과 직선사이의 거리를 구하는 문제로 바뀌고, 그러므로 벡터의 정사영을 사용하여 간단히 해결할 수 있다. 따라서  $ax + b$ 를 가장 작게 만드는  $a$ 를 구하면 아래와 같다.

$$a = -\frac{x \bullet b}{x \bullet x}$$

다음으로  $a$ 에 관하여  $v$ 를 풀면

$$\begin{aligned} a(2v - b_x - b_y) &= x + y - b_x - b_y \\ \therefore v &= \frac{x + y - b_x - b_y}{2a} + \frac{b_x + b_y}{2} \end{aligned}$$

따라서 클릭하면 색이 달라지는 이미지를 만드는 공식을 얻었다. 위 식에서는  $a < 0$ 이거나  $a > 1$ 일 수 있는 바, 실제 코딩할 때에는 각 변수의 범위를 제한하였다.

그런데 위의 방법은  $a$ 에 대해 근사하고,  $v$ 를  $a$ 에 맞추었다. 그러나 어쩌면  $v$ 와  $a$ 를 동시에 근사하면 더 좋은 결과를 얻을지도 모른다. 만약 근사를 할 때, 오차함수를 아래와 같이 주면 어떻게 되는가?

$$err = (x - (v - b_x)a - b_x)^2 + (y - (v - b_y)a - b_y)^2$$

이 오차함수를 최소화하는 것은 5차원 공간상의 4차원 표면  $err = f(a, v_r, v_g, v_b)$ 에서 국소점을 찾는 문제이다. 식을 치환하고 정리하면 아래와 같은 식의 국소점을 구하는 문제가 된다.

$$y = \left( \frac{A}{a} - v + B \right)^2 = \frac{A^2}{a^2} + v^2 + B^2 + 2 \left( -\frac{Av}{a} + \frac{AB}{a} - vB \right)$$

$$\nabla y = \left\langle -2A^2 a^{-3} + 2A(v-B)a^{-2}, 2v_x 2 \left( -\frac{A_x v_x}{a} + v B_x \right), \dots, \dots \right\rangle$$

그러나 이를 직접 계산하는 것은 너무 힘들기 때문에, 매트랩의 Symbolic 기능을 사용하여 계산하였다.

$R = (bx2^2*x1 + bx3^2*x1 + 2^*by1^2*x1 + by2^2*x1 + by3^2*x1 + 2^*bx1^2*y1 + bx2^2*y1 + bx3^2*y1 + by2^2*y1 + by3^2*y1 - bx1^2*bx2*x2 - bx1^2*bx3*x3 - 2^*bx1^2*by1*x1 + bx1^2*by2*x2 - bx2^2*by1*x2 - 2^*bx2^2*bx1*x1 + bx1^2*by3*x3 - bx3^2*by1*x3 - 2^*bx3^2*bx1*x1 + by1^2*bx2*x2 + by1^2*bx3*x3 + bx1^2*bx2^2*y1 + bx1^2*bx3^2*y3 - 2^*bx1^2*by1*y1 - bx1^2*by2*y2 + bx2^2*by1*y2 - 2^*bx2^2*by2*y1 - bx1^2*by3*y3 + bx3^2*by1*y3 - 2^*bx3^2*by3*y1 - by1^2*bx1*y2 - by1^2*bx3*y3)/(2*(by1*x1 - 2^*bx2^2*by2 - 2^*bx3^2*by3 - bx1*x1 - bx2*x2 - bx3*x3 - 2^*bx1^2*by1 + by2*x2 + by3*x3 + bx1*y1 + bx2*y2 + bx3*y3 - bx1*y1 - by2*y2 - by3*y3 + bx1*x2 + bx2*x2 + bx3*x2 + by1*x2 + by2*x2 + by3*x2))$ ,  
 $G = (bx1^2*x2 + bx3^2*x2 + by1^2*x2 + 2^*by2^2*x2 + by3^2*x2 + bx1^2*y2 + 2^*bx2^2*y2 + bx3^2*y2 + by1^2*y2 + by3^2*y2 - bx1^2*bx2*x1 - bx2^2*bx3*x3 - 2^*bx1^2*by1*x2 - bx1^2*by2*x1 + bx2^2*by1*x1 - 2^*bx2^2*by2*x2 + bx2^2*by3*x3 - bx3^2*by2*x3 - 2^*bx3^2*by3*x2 + by1^2*bx1*x1 + by2^2*bx3*x3 + bx1^2*bx2^2*y1 + bx2^2*bx3^2*y3 - 2^*bx1^2*by1*y2 + bx1^2*by2*y1 - bx2^2*by1*y1 - 2^*bx2^2*by2*y2 - bx2^2*by3*y3 + bx3^2*by2*y3 - 2^*bx3^2*by3*y1 - by1^2*bx1*y2 - by2^2*bx3*y3)/(2*(by1*x1 - 2^*bx2^2*by2 - 2^*bx3^2*by3 - bx1*x1 - bx2*x2 - bx3*x3 - 2^*bx1^2*by1 + by2*x2 + by3*x3 + bx1*y1 + bx2*y2 + bx3*y3 - bx1*y1 - by2*y2 - by3*y3 + bx1*x2 + bx2*x2 + bx3*x2 + by1*x2 + by2*x2 + by3*x2))$ ,  
 $B = (bx1^2*x3 + bx2^2*x3 + by1^2*x3 + by2^2*x3 + 2^*by3^2*x3 + bx1^2*y3 + bx2^2*y3 + bx3^2*y3 + by1^2*y3 + by2^2*y3 - bx1^2*bx3*x1 - bx2^2*bx3*x2 - 2^*bx1^2*by1*x3 - bx1^2*by3*x1 + bx3^2*by1*x1 - 2^*bx2^2*by2*x3 - bx2^2*by3*x2 + bx3^2*by2*x2 - 2^*bx3^2*by3*x3 + by1^2*bx3*y1 + by2^2*bx3*x2 + bx1^2*bx3^2*y1 + bx2^2*bx3^2*y2 - 2^*bx1^2*by1*y3 + bx1^2*by3*y1 - bx3^2*by1*y1 - 2^*bx2^2*by2*y3 + bx2^2*by3*y2 - bx3^2*by2*y3 - 2^*bx3^2*by3*y1 - by1^2*bx3*y1 - by2^2*bx3*y2)/(2*(by1*x1 - 2^*bx2^2*by2 - 2^*bx3^2*by3 - bx1*x1 - bx2*x2 - bx3*x3 - 2^*bx1^2*by1 + by2*x2 + by3*x3 + bx1*y1 + bx2*y2 + bx3*y3 - bx1*y1 - by2*y2 - by3*y3 + bx1*x2 + bx2*x2 + bx3*x2 + by1*x2 + by2*x2 + by3*x2))$ ,  
 $A = (bx1*x1 - 2^*bx2^2*by2 - 2^*bx3^2*by3 - bx1*x1 - bx2*x2 - bx3*x3 - 2^*bx1^2*by1 + by2*x2 + by3*x3 + bx1*y1 + bx2*y2 + bx3*y3 - by1*y1 - by2*y2 - by3*y3 + bx1^2 + bx2^2 + bx3^2 + by1*x2 + by2*x2 + by3*x2)/(bx1^2 - 2^*bx1^2*by1 + bx2*x2 - 2^*bx2^2*by2 + bx3^2 - 2^*bx3^2*by3 + by1^2 + by2^2 + by3^2)$

### Matlab을 사용하여 계산한 결과

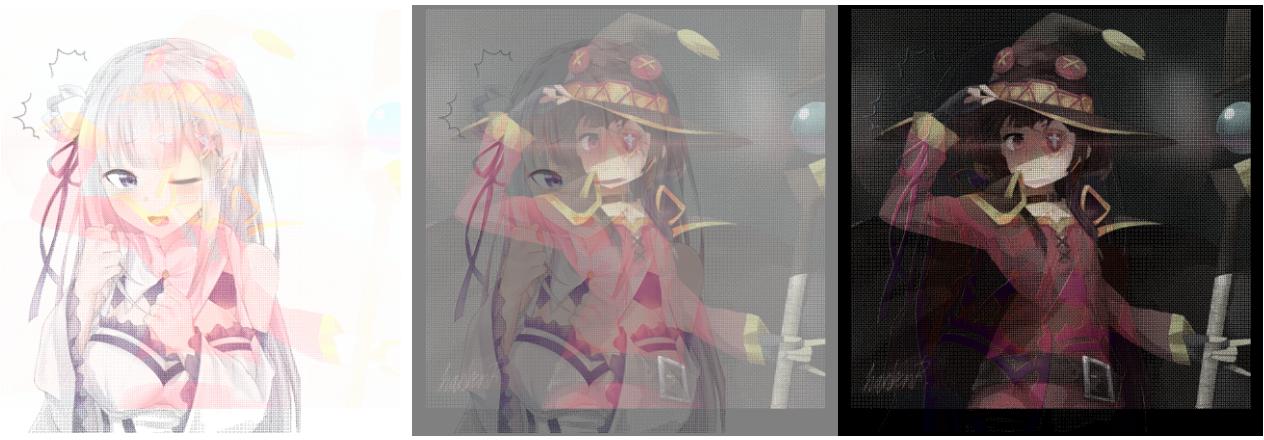
그러나 이것 역시 너무 계산 시간을 많이 소모하므로,  $b_x = 255, b_y = 0$ 으로 가정하고 계산해보았다. 결과는 아래와 같다.

$$\begin{aligned} R &= (130050*x1 - 65025*x2 - 65025*x3 + 260100*y1 + 65025*y2 + 65025*y3)/(2*(255*y1 - 255*x2 - 255*x3 - 255*x1 + 255*y2 + 255*y3 + 195075)) \\ G &= (130050*x2 - 65025*x1 - 65025*x3 + 65025*y1 + 260100*y2 + 65025*y3)/(2*(255*y1 - 255*x2 - 255*x3 - 255*x1 + 255*y2 + 255*y3 + 195075)) \\ B &= (130050*x3 - 65025*x2 - 65025*x1 + 65025*y1 + 65025*y2 + 260100*y3)/(2*(255*y1 - 255*x2 - 255*x3 - 255*x1 + 255*y2 + 255*y3 + 195075)) \\ A &= y1/3 - x2/3 - x3/3 - x1/3 + y2/3 + y3/3 + 255 \end{aligned}$$

$b_x = 255, b_y = 0$ 으로 가정하고 계산한 결과

위 두 식 모두  $\nabla err = 0$ 이 되는 RGBA에 대한 유일한 해가 아니다. 그러나 다른 해는 R,G,B,A값들 중 두 개 이상이 0인 등, 우리가 원하는 결과가 아니었다. 이는 아마도 해가  $b_x, b_y$ 에 대하여 대칭적이지 않거나,  $\nabla err = 0$ 이지만 국소가 아니라 안장점이었을 것이다.

위 두 방법을 모두 python으로 구현하고 확인해본 결과, 육안으로 구별할 수 있을 만큼 유의미한 차이를 보이지 않았다. 아래 두 이미지는 위의 알고리즘으로 생성한 서로 같은 이미지이다.



완벽한 이미지 변환을 구현한 것은 아니나, 충분히 유의미하고 유색인 이미지에서도 작동하는 알고리즘을 구현하였다. 예제 이미지를 드리