Image color replacement
with
Numerical optimization

Image color replacement with numerical optimization

Informal problem statement

Input:

- Image
- User-provided colors to replace
- User-provided colors to replace **with**
- User-provided colors that shouldn’t change
- (Optimization-specific parameters)

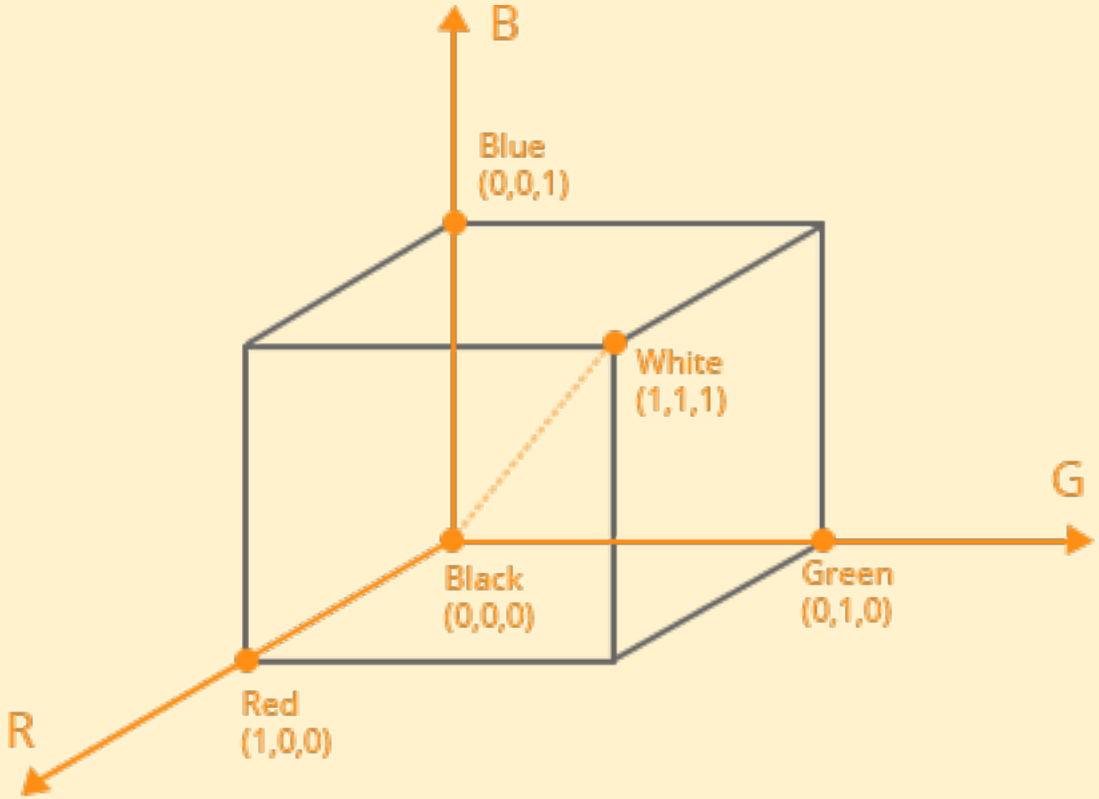
Output: Image with replaced colors

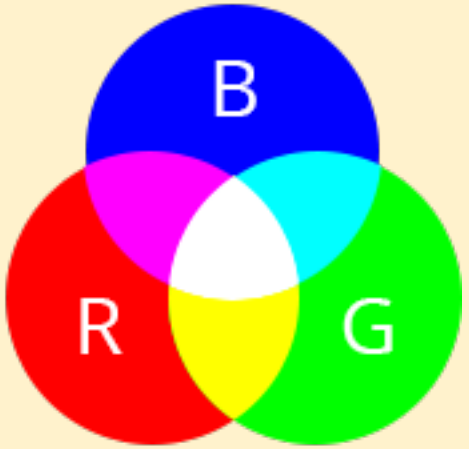
Digital image formats

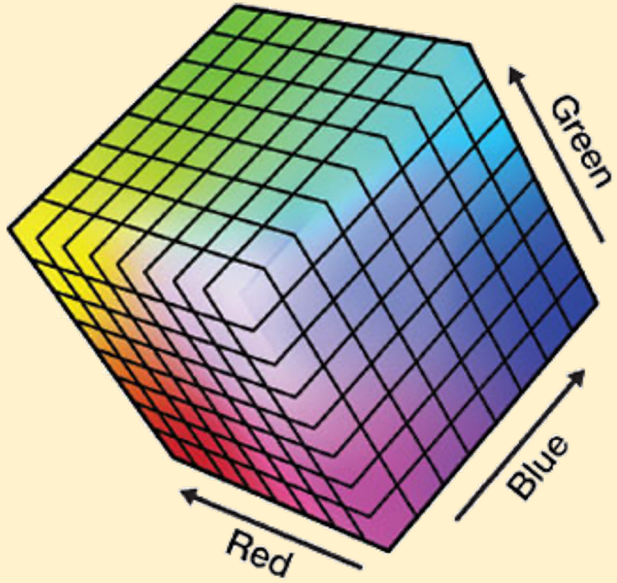
RGB color space

Digital images are (usually) 2D tables of pixels, where each pixel is specified in the RGB color space (and sometimes additional alpha channel for transparency).

* There are other color spaces for various editing / processing purposes, e.g. HSV, LA*B*, …







Problem formulation

Inputs / constants

$I \in \mathbb{R}^{n\times 3}$ - Flattened image

$C_1 \in \mathbb{R}^{m\times 3}$ - Colors to replace (including fixed colors)

$C_2 \in \mathbb{R}^{m\times 3}$ - Color to replace with (including fixed colors)

Variables

$T \in \mathbb{R}^{3\times k}$

$B \in \mathbb{R}^{k\times 3}$

$N \in \mathbb{R}^{k\times 3}$

Desired transformation: $\sigma(\sigma(IT)N)$

Problem

Minimize:

$$\sum |B| + \sum |N| + \sum |\sigma(IT)|$$

Subject to:

$$\sigma(\sigma(IT)B) = I$$
$$\sigma(\sigma(C_1T)N) = C_2$$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Intuition

- First represent the image in a (sparse) “latent color space”
- Then find a transformation that performs the desired replacement
- Nonlinear transformations (sigmoid function)

But…

Not convex, strong duality doesn’t hold; doesn’t admit to an existing optimal technique.

Penalty method alternative problem formulation

Inputs / constants

$I \in \mathbb{R}^{n \times 3}$

$C_1 \in \mathbb{R}^{m \times 3}$

$C_2 \in \mathbb{R}^{m \times 3}$

Variables

$T \in \mathbb{R}^{3 \times k}$

$B \in \mathbb{R}^{k \times 3}$

$N \in \mathbb{R}^{k \times 3}$

$\theta = T, B, N$

Add to the original minimization objective a weighted term penalizing constraint violation

Minimize "loss" function

$$J(\theta) = \frac{1}{\dots} \sum |B| + \frac{1}{\dots} \sum |N| + \frac{1}{\dots} \sum |\sigma(IT)| + \lambda[\frac{1}{I_{rows}} \sum_{i=1}^{I_{rows}} ||(I - \sigma(\sigma(IT)B))_i||_2 + \frac{1}{C_{2;rows}} \sum_{i=1}^{C_{2;rows}} ||(C_2 - \sigma(\sigma(C_1T)N))_i||_2] = \sum_{i=1}^{5} J_i$$

Optimization approach - BFGS

- Find analytical gradients for the problem
- Implement BFGS
  - Use Wolfe conditions in line search

Problem gradients

5 loss components, 3 parameter “families” (T, B, N)

$$J_1(\theta) = \frac{1}{\dots}|B|$$
$$\frac{\partial J_1}{\partial N} = \frac{\partial \dot{\dot{J}}_1}{\partial T} = 0$$
$$\frac{\partial J_1}{\partial B} = \frac{sgn(B)}{\dots}$$

$$J_2(\theta) = \frac{1}{\dots}|N|$$
$$\frac{\partial J_2}{\partial B} = \frac{\partial \dot{\dot{J}}_2}{\partial T} = 0$$
$$\frac{\partial J_2}{\partial N} = \frac{sgn(N)}{\dots}$$

$$J_3(\theta) = \frac{1}{\dots}|\sigma(IT)|$$
$$\frac{\partial J_3}{\partial B} = \frac{\partial \dot{\dot{J}}_3}{\partial N} = 0$$

$$A_I = \sigma(IT)$$

$$\frac{\partial J_3}{\partial T} = \frac{\partial J_3}{\partial A_I} \cdot \frac{\partial \sigma(IT)}{\partial IT} \cdot \frac{\partial IT}{\partial T} = I^T \cdot sgn(\sigma(IT)) \cdot \sigma(IT)(1 - \sigma(IT))$$

Problem gradients

5 loss components, 3 parameter “families” (T, B, N)

$$J_4(\theta) = \frac{1}{\cdots} \sum_{i=1}^{\cdots} ||(I - \sigma(\sigma(IT)B))_i||_2 \qquad y = \sigma(A_I \cdot B) = \sigma(\sigma(IT)B)$$

$$\boxed{\frac{\partial J_4}{\partial N} = 0}$$

$$\frac{\partial J_4}{\partial T} = \frac{\partial J_4}{\partial y} \cdot \frac{\partial y}{\partial T}$$

$$\frac{\partial J_4}{\partial y_{*,0}} = \frac{1}{\cdots} \frac{\partial \sqrt{((I_{*,0} - y_{*,0})^2 + a)}}{\partial y_{*,0}}$$

$$f(x) = \sqrt{x^2 + a}$$

$$\frac{df}{dx} = \frac{1}{2}(x^2 + a)^{-\frac{1}{2}} \cdot 2x = \frac{x}{\sqrt{x^2 + a}} = \frac{x}{f(x)}$$

Problem gradients

5 loss components, 3 parameter “families” (T, B, N)

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$$ (Classical NN derivation)

$$\frac{\partial y_{*,0}}{\partial T_{*,0}} = \frac{\partial y_{*,0}}{\partial A_{I*,0}} \cdot \frac{\partial A_{I*,0}}{\partial T_{*,0}}$$

$$\frac{\partial y_{*,0}}{\partial A_{I*,0}} = \frac{\partial \sigma(A_{I*,0}B_{0,0})}{\partial A_{I*,0}B_{0,0}} \cdot \frac{\partial A_{I*,0}B_{0,0}}{\partial A_{I*,0}} = B_{0,0} \cdot \sigma(A_{I*,0} \cdot B_{0,0}) \cdot (1 - \sigma(A_{I*,0} \cdot B_{0,0}))$$

$$\frac{\partial A_{I*,0}}{\partial T_{*,0}} = \frac{\partial \sigma(I \cdot T_{*,0})}{\partial I \cdot T_{*,0}} \cdot \frac{\partial I \cdot T_{*,0}}{\partial T_{*,0}} = \sigma(I \cdot T_{*,0}) \cdot (1 - \sigma(I \cdot T_{*,0})) \cdot I$$

Problem gradients

5 loss components, 3 parameter “families” (T, B, N)

$$\frac{\partial J_4}{\partial B} = \frac{\partial J_4}{\partial y} \cdot \frac{\partial y}{\partial B}$$

$$\frac{\partial y_{*,0}}{\partial B_{*,0}} = \frac{\partial \sigma(A_{I*,0} \cdot B_{*,0})}{\partial A_{I*,0} \cdot B_{*,0}} \cdot \frac{\partial A_{I*,0} \cdot B_{*,0}}{\partial B_{*,0}} = \sigma(A_{I*,0} \cdot B_{*,0}) \cdot (1 - \sigma(A_{I*,0} \cdot B_{*,0})) \cdot A_{I*,0}$$

Problem gradients

5 loss components, 3 parameter “families” (T, B, N)

Derivation of J_5 is similar to J_4 but replacing I with C_1, C_2 and B with N.

Implementation

- BFGS, penalty method, line search – straightforward (most of the work was getting the gradients right)
- Using `numba.jit` for performance improvement
- Using YML file for configuration (user “input” + optimization parameters)

Runtime ~ 04:30 minutes on local machine, flowers image (1600x900) with scale 0.3 for 100 iterations (didn’t stop early)

```yaml
input_image_path: "sample_images/ice_cream.jpeg"
output_image_path: "results/ice_cream_2.jpeg"
from_colors:
  - [251, 201, 142]
to_colors:
  - [109, 62, 35]
fixed_colors:
  - [171, 94, 50]
  - [181, 150, 145]
  - [214, 202, 204]
  - [152, 0, 10]
  - [160, 159, 157]
  - [157, 147, 138]
  - [194, 184, 174]
  - [213, 178, 148]
  - [155, 135, 170]  # Purple ice cream
  - [8, 207, 248]  # Teal ice cream
  - [138, 246, 222]  # Green ice cream
  - [250, 197, 223]  # Pink ice cream
latent_color_components_k: 12
optimization_scale: 0.25
wolfe_conditions:
  c1: 0.3
  c2: 0.8
penalty_weights: [50, 100]
bfgs_max_iterations: 100
```


THE
END