Linear Constraint Construction for Gamut-Constrained Color Transformation

We aim to optimize a color transformation matrix $T \in \mathbb{R}^{3\times 3}$ such that when it is applied to an RGB contrast image, the resulting RGB values remain within gamut: $[0, 1]$. We want the same to be true for a dichromatic rendering of the image. This leads to a linear inequality constraint of the form:

$$Ax \leq b$$

where $x = \mathrm{vec}(T) \in \mathbb{R}^{9\times 1}$ is the flattened 3×3 transformation matrix.
The trichromatic RGB contrast image is represented in cal format as:

$$\texttt{triRGBContrastCalFormat} = \begin{bmatrix} r_1 & r_2 & \cdots & r_n \\ g_1 & g_2 & \cdots & g_n \\ b_1 & b_2 & \cdots & b_n \end{bmatrix} \in \mathbb{R}^{3\times n}$$

Each column corresponds to the RGB contrast values of one pixel i, i.e.,

$$\texttt{triRGBContrastCalFormat}(:, i) = \begin{bmatrix} r_i \\ g_i \\ b_i \end{bmatrix}$$

Constraint Matrix for Trichromatic Rendering

The full trichromatic constraint matrix $A_{\mathrm{tri}} \in \mathbb{R}^{3n\times 9}$ is constructed by stacking per-pixel blocks $A_i \in \mathbb{R}^{3\times 9}$. Each block is of the form:

$$A_i = \begin{bmatrix} r_i & 0 & 0 & g_i & 0 & 0 & b_i & 0 & 0 \\ 0 & r_i & 0 & 0 & g_i & 0 & 0 & b_i & 0 \\ 0 & 0 & r_i & 0 & 0 & g_i & 0 & 0 & b_i \end{bmatrix}$$

The full matrix for n pixels is:

$$A_{\mathrm{tri}} = \begin{bmatrix} r_1 & 0 & 0 & g_1 & 0 & 0 & b_1 & 0 & 0 \\ 0 & r_1 & 0 & 0 & g_1 & 0 & 0 & b_1 & 0 \\ 0 & 0 & r_1 & 0 & 0 & g_1 & 0 & 0 & b_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ r_i & 0 & 0 & g_i & 0 & 0 & b_i & 0 & 0 \\ 0 & r_i & 0 & 0 & g_i & 0 & 0 & b_i & 0 \\ 0 & 0 & r_i & 0 & 0 & g_i & 0 & 0 & b_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ r_n & 0 & 0 & g_n & 0 & 0 & b_n & 0 & 0 \\ 0 & r_n & 0 & 0 & g_n & 0 & 0 & b_n & 0 \\ 0 & 0 & r_n & 0 & 0 & g_n & 0 & 0 & b_n \end{bmatrix} \in \mathbb{R}^{3n\times 9}$$

Let the transformation matrix $T \in \mathbb{R}^{3\times 3}$ be:

$$T = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix}$$

and let its vectorized form be:

$$x = \mathrm{vec}(T) = \begin{bmatrix} T_{11} \\ T_{21} \\ T_{31} \\ T_{12} \\ T_{22} \\ T_{32} \\ T_{13} \\ T_{23} \\ T_{33} \end{bmatrix} \in \mathbb{R}^{9\times 1}$$

Let the RGB contrast vector for pixel i be the following:

$$\mathbf{c}_i = \begin{bmatrix} r_i \\ g_i \\ b_i \end{bmatrix} \in \mathbb{R}^{3\times 1}$$

So when we apply the transformation matrix to this RGB pixel we get the following new RGB pixel:

$$T \cdot \mathbf{c}_i = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \begin{bmatrix} r_i \\ g_i \\ b_i \end{bmatrix} = \begin{bmatrix} r_iT_{11} + g_iT_{12} + b_iT_{13} \\ r_iT_{21} + g_iT_{22} + b_iT_{23} \\ r_iT_{31} + g_iT_{32} + b_iT_{33} \end{bmatrix}$$

Let's show that we get the same thing if we use the new A matrix and a strung out version of the transformation matrix. Define $A_i \in \mathbb{R}^{3\times 9}$ as:

$$A_i = \begin{bmatrix} r_i & 0 & 0 & g_i & 0 & 0 & b_i & 0 & 0 \\ 0 & r_i & 0 & 0 & g_i & 0 & 0 & b_i & 0 \\ 0 & 0 & r_i & 0 & 0 & g_i & 0 & 0 & b_i \end{bmatrix}$$

$$A_i \cdot x = \begin{bmatrix} r_i & 0 & 0 & g_i & 0 & 0 & b_i & 0 & 0 \\ 0 & r_i & 0 & 0 & g_i & 0 & 0 & b_i & 0 \\ 0 & 0 & r_i & 0 & 0 & g_i & 0 & 0 & b_i \end{bmatrix} \begin{bmatrix} T_{11} \\ T_{21} \\ T_{31} \\ T_{12} \\ T_{22} \\ T_{32} \\ T_{13} \\ T_{23} \\ T_{33} \end{bmatrix} = \begin{bmatrix} r_iT_{11} + g_iT_{12} + b_iT_{13} \\ r_iT_{21} + g_iT_{22} + b_iT_{23} \\ r_iT_{31} + g_iT_{32} + b_iT_{33} \end{bmatrix}$$

$$T \cdot \mathbf{c}_i = A_i \cdot x$$

This shows that applying T directly to each pixel's RGB vector is exactly equivalent to multiplying the flattened vector $x = \mathrm{vec}(T)$ with the corresponding matrix block A_i. For all pixels, it looks like this:

$$A_{\text{tri}} \cdot x = \begin{bmatrix} T \cdot \begin{bmatrix} r_1 \\ g_1 \\ b_1 \end{bmatrix} \\ T \cdot \begin{bmatrix} r_2 \\ g_2 \\ b_2 \end{bmatrix} \\ \vdots \\ T \cdot \begin{bmatrix} r_n \\ g_n \\ b_n \end{bmatrix} \end{bmatrix} \in \mathbb{R}^{3n\times 1}$$

To ensure that all transformed RGB values remain within the gamut $[0, 1]$, we enforce the inequality:

$$\mathbf{0} \leq A_{\text{tri}} \cdot x \leq \mathbf{1}$$

The less than 1 part of the inequality is easy enough to understand. It's as simple as saying "A times x (or T) must be less than 1 everywhere"

$$A_{\text{tri}} \cdot x \leq \mathbf{1}$$

However, the `fmincon` linear constraint only allows constraints of the form:

$$Ax \leq b$$

So how do we enforce the lower bound:

$$A_{\text{tri}} \cdot x \geq \mathbf{0} \quad ?$$

We use a simple trick: multiply both sides of the inequality by -1, which flips the inequality sign:

$$A_{\text{tri}} \cdot x \geq \mathbf{0} \quad \Longleftrightarrow \quad -A_{\text{tri}} \cdot x \leq \mathbf{0}$$

This converts the lower bound into a standard form suitable for the optimizer.

But wait!! Remember that the RGB values in the A matrix are contrast values, not regular RGB values. It is the regular RGB values that need to stay between 0 and 1, not the contrast RGB.

Let the gray RGB value be defined as:

$$\text{gray} = \begin{bmatrix} g_R \\ g_G \\ g_B \end{bmatrix} \in \mathbb{R}^{3\times 1}$$

Then each input pixel contrast vector is:

$$\mathbf{c}_i = \frac{\text{pixel}_i - \text{gray}}{\text{gray}}$$

We build the constraint matrix $A_{\text{contrast}} \in \mathbb{R}^{3n\times 9}$ from these normalized contrast values using:

$$A_i = \begin{bmatrix} c_{iR} & 0 & 0 & c_{iG} & 0 & 0 & c_{iB} & 0 & 0 \\ 0 & c_{iR} & 0 & 0 & c_{iG} & 0 & 0 & c_{iB} & 0 \\ 0 & 0 & c_{iR} & 0 & 0 & c_{iG} & 0 & 0 & c_{iB} \end{bmatrix} \quad \text{where } \mathbf{c}_i = \begin{bmatrix} c_{iR} \\ c_{iG} \\ c_{iB} \end{bmatrix}$$

The output of the transformation is:

$$\mathbf{y} = A_{\text{contrast}} \cdot T \quad \text{where } T \in \mathbb{R}^{9\times 1}$$

Since $\mathbf{y}$ contains RGB *contrast* values, we must convert it back into RGB intensity space before enforcing gamut constraints. The full transformation to intensity is:

$$\text{RGB} = (A_{\text{contrast}} \cdot T) \circ \text{gray} + \text{gray}$$

To enforce:

$$0 \leq \text{RGB} \leq 1$$

we substitute the expression above and get:

$$0 \leq (A_{\text{contrast}} \cdot T) \circ \text{gray} + \text{gray} \leq 1$$

Subtract gray on all sides:

$$-\text{gray} \leq (A_{\text{contrast}} \cdot T) \circ \text{gray} \leq 1 - \text{gray}$$

Now divide elementwise by gray:

$$-\frac{\text{gray}}{\text{gray}} \leq A_{\text{contrast}} \cdot T \leq \frac{1-\text{gray}}{\text{gray}}$$

Which simplifies to:

$$-\mathbf{1} \leq A_{\text{contrast}} \cdot T \leq \frac{1-\text{gray}}{\text{gray}}$$

To cast this in the standard linear inequality form required by `fmincon`, we split and negate the inequalities:

$$A_{\text{contrast}} \leq \frac{1-\text{gray}}{\text{gray}}$$

$$-A_{\text{contrast}} \leq 1$$

Then stack - subscript 'tri' is for trichromat. We will build a constraint matrix both for the trichromat (below) as well as a constraint that ensures the dichromat rendering is in gamut (next page).

$$A_{\text{tri}} = \begin{bmatrix} A_{\text{contrast}} \\ -A_{\text{contrast}} \end{bmatrix}, \quad b_{\text{tri}} = \begin{bmatrix} \frac{1-\text{gray}}{\text{gray}} \\ \mathbf{1} \end{bmatrix}$$

If gray is just all 0.5, then

$$A_{\text{tri}} = \begin{bmatrix} A_{\text{contrast}} \\ -A_{\text{contrast}} \end{bmatrix}, \quad b_{\text{tri}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

which guarantees that the final RGB values after transformation fall within the valid gamut $[0, 1]$.

Constraint Matrix for Dichromatic Rendering

In addition to ensuring that the transformed RGB values remain in gamut for a trichromatic observer, we also wish to ensure that the simulated RGB values as perceived by a dichromat (e.g., a protanope or deuteranope) also remain within the valid display gamut $[0, 1]$. To achieve this, we simulate how a dichromat perceives the image by applying a linear transformation matrix to the RGB contrast values.
Let $M_{\text{tri}\to\text{di}} \in \mathbb{R}^{3\times3}$ be the matrix that simulates dichromatic perception by mapping trichromatic LMS contrast values to dichromatic LMS contrast values.
Let $M_{\text{rgb}\to\text{cones}} \in \mathbb{R}^{3\times3}$ be the display matrix that converts RGB values into cone excitations.
We define the matrix that maps RGB contrast (trichromatic) to RGB contrast (as perceived by a dichromat) as:

$$M_{\text{conesC}\to\text{rgbC}} = \text{diag}\left(\frac{1}{\text{grayRGB}}\right) \cdot M_{\text{rgb}\to\text{cones}}^{-1} \cdot \text{diag}(\text{grayLMS})$$

Then, the full transformation matrix is:

$$M_{\text{rgbCtri}\to\text{rgbCdi}} = M_{\text{conesC}\to\text{rgbC}} \cdot M_{\text{tri}\to\text{di}} \cdot M_{\text{conesC}\to\text{rgbC}}^{-1} \in \mathbb{R}^{3\times3}$$

$$M_{\text{rgbCtri}\to\text{rgbCdi}} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}_{M_{\text{conesC}\to\text{rgbC}}} \cdot \underbrace{\begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix}}_{M_{\text{triConesC}\to\text{diConesC}}} \cdot \underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}_{M_{\text{rgbC}\to\text{conesC}}}$$

This matrix acts in contrast space, transforming trichromat RGB contrast into the RGB contrast a dichromat would perceive. So if you multiply this with a contrast rgb image (the rgb image on the right in cal format) then out you get a contrast rgb image for a deuteranope
To apply this same transformation to every pixel in the image independently, we use the Kronecker product:

$$M_{\text{big}} = I_n \otimes M_{\text{rgbCtri}\to\text{rgbCdi}} \in \mathbb{R}^{3n\times 3n}$$

where $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix for n pixels. The Kronecker product builds a block-diagonal matrix that applies M_{all} to each pixel's 3-element RGB contrast vector.
This expands to a block-diagonal matrix where each block is M_{all}, one per pixel.
For example, when $n = 3$, the matrix becomes:

$$M_{\text{big}} = \begin{bmatrix} M_{\text{rgbCtri}\to\text{rgbCdi}} & 0 & 0 \\ 0 & M_{\text{rgbCtri}\to\text{rgbCdi}} & 0 \\ 0 & 0 & M_{\text{rgbCtri}\to\text{rgbCdi}} \end{bmatrix} \in \mathbb{R}^{9\times 9}$$

where each $M_{\text{all}} \in \mathbb{R}^{3\times 3}$ operates on one pixel's 3-element RGB contrast vector.
If we denote:

$$M_{\text{rgbCtri}\to\text{rgbCdi}} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

then M_{big} when $n = 3$ is:

$$M_{\text{big}} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{11} & a_{12} & a_{13} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{31} & a_{32} & a_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{11} & a_{12} & a_{13} \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

This structure ensures that each pixel's contrast vector is independently transformed by the same M_{all}, preserving pixelwise independence in the dichromatic simulation.
We already constructed the trichromatic constraint matrix:

$$A_{\text{tri}} \in \mathbb{R}^{3n\times 9}$$

To simulate the RGB contrast as perceived by a dichromat, we left-multiply this matrix by M_{big}, giving:

$$A_{\text{di}} = M_{\text{big}} \cdot A_{\text{tri}} \in \mathbb{R}^{3n\times 9}$$

So the full dichromat constraint matrix is:

$$A_{\text{di}} = \begin{bmatrix} M_{\text{rgbCtri}\to\text{rgbCdi}} \cdot A_1 \\ M_{\text{rgbCtri}\to\text{rgbCdi}} \cdot A_2 \\ \vdots \\ M_{\text{rgbCtri}\to\text{rgbCdi}} \cdot A_n \end{bmatrix} \in \mathbb{R}^{3n\times 9}$$

In this formulation, the matrix-vector product:

$$A_{\text{di}} \cdot x = M_{\text{big}} \cdot (A_{\text{tri}} \cdot x)$$

represents the RGB contrast output seen by a dichromat after:

1. Applying the color transformation T,
2. Simulating dichromatic perception via M_{all},
3. And combining both in a linear constraint of the form $A_{\text{di}} \cdot x \leq b$.

To enforce both upper and lower bounds on the simulated dichromatic RGB values, we construct:

$$A_{\text{di}} = \begin{bmatrix} A_{\text{di}} \\ -A_{\text{di}} \end{bmatrix} = \begin{bmatrix} M_{\text{big}} \cdot A_{\text{tri}} \\ -M_{\text{big}} \cdot A_{\text{tri}} \end{bmatrix} \in \mathbb{R}^{6n \times 9}$$

To preserve the same output range $[0, 1]$ in RGB space for the dichromatic rendering, we use the same bounds:

$$b_{\text{di}} = \begin{bmatrix} \mathbf{1}_{3n \times 1} \\ \mathbf{1}_{3n \times 1} \end{bmatrix} \in \mathbb{R}^{6n \times 1}$$

This enforces that the simulated dichromatic RGB values (obtained after applying the transformation T, followed by M_{all}) also fall within gamut.
The matrix–vector product $A_{\text{di}} \cdot x$ corresponds to applying the color transform T, then simulating dichromatic perception, and finally checking whether the resulting RGB contrast values remain in valid range. This constraint ensures that the optimization produces RGB outputs that are perceptually valid for both trichromats and dichromats.

Final Combined Constraint

We now combine both constraints:

$$A_{\text{total}} = \begin{bmatrix} A_{\text{tri}} \\ A_{\text{di}} \end{bmatrix} \in \mathbb{R}^{12n \times 9}, \quad b_{\text{total}} = \begin{bmatrix} b_{\text{tri}} \\ b_{\text{di}} \end{bmatrix} \in \mathbb{R}^{12n \times 1}$$

Our optimization constraint is then:

$$A_{\text{total}} \cdot x \leq b_{\text{total}}$$

This ensures that both the trichromatic and simulated dichromatic renderings of the image remain within the RGB gamut.