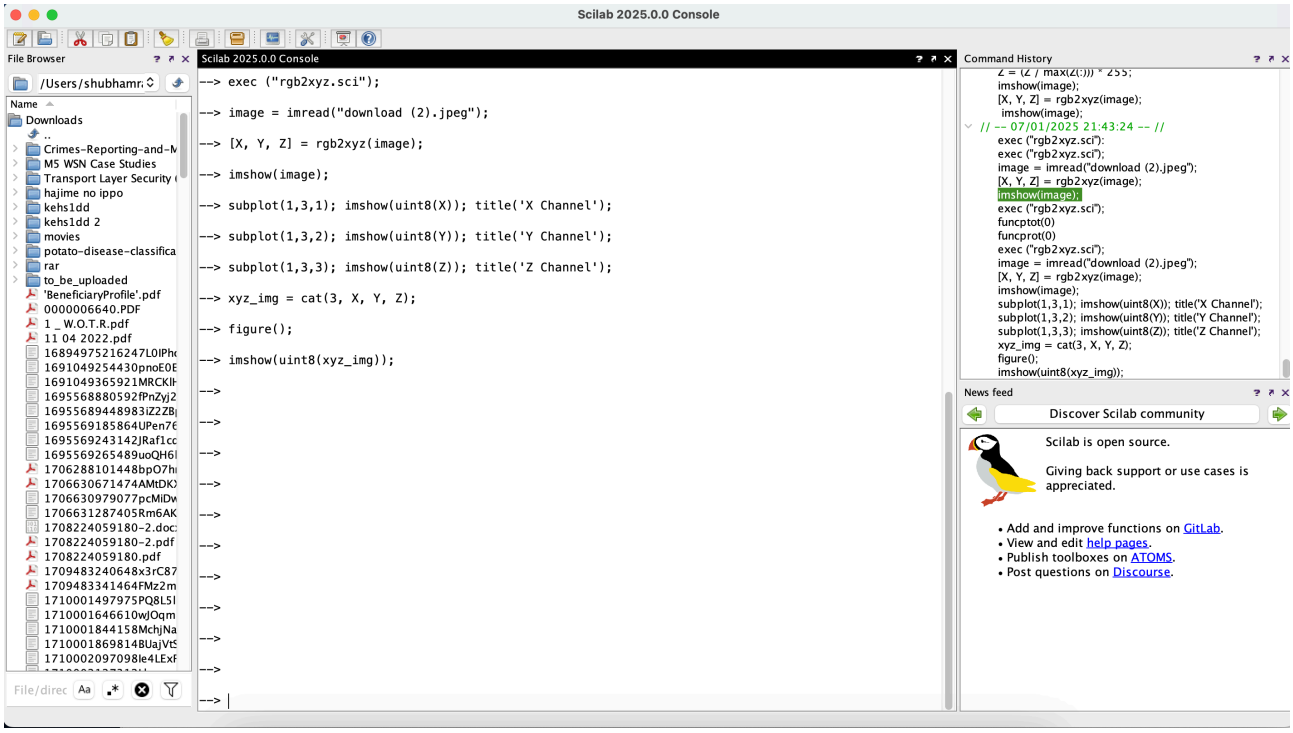
Overview

The rgb2xyz function converts an RGB image or separate RGB channel inputs to the XYZ color space using the D65 illuminant. This transformation is commonly used in color science as it is device-independent and serves as a basis for various other color models like LAB and LUV. This is an scilab implementation of octave's function rgb2xyz.

Purpose:

- Converts RGB color values to the XYZ color space.
- Supports conversion of RGB images (3D arrays) or individual RGB channels.
- Includes gamma correction to ensure proper color mapping.

Calling Sequence:



Defining and executing function file:

exec ("rgb2xyz.sci");
This line executes the rgb2xyz.sci script, which defines the rgb2xyz function in Scilab. It loads the function into memory so that it can be used in the following commands.

Loading the image:

image = imread("image.jpeg");
This function reads an image file ("image.jpeg") and loads it into the image variable. The imread function returns the image as array where the dimensions are height x width x 3, corresponding to the red, green, and blue channels.

Function Signature

function [X, Y, Z]=rgb2xyz(varargin)

Input Parameters

- varargin: A variable-length input argument list that can take either:
  - A single RGB image (3D array with dimensions [height, width, 3])
  - Three separate 2D matrices representing R, G, and B channels

Output Parameters

- X: The X channel of the converted image in XYZ color space
- Y: The Y channel of the converted image in XYZ color space
- Z: The Z channel of the converted image in XYZ color space

Function Breakdown

Step 1: Determine Input Format

The function checks the number of input arguments (nargs):

- If nargs == 1, the input is assumed to be a single RGB image. The function extracts the three color channels.
- If nargs == 3, the input is treated as three separate grayscale images representing R, G, and B components.
- If neither condition is met, an error is raised.

Step 2: Input Validation

```
R = double(max(0, min(255, R)));
G = double(max(0, min(255, G)));
 B = double(max(0, min(255, B)));
```

Each pixel value is clamped between 0 and 255 to ensure valid intensity values. This step prevents overflow and underflow errors when processing the image.

Step 3: Normalize RGB Values

```
R = R / 255;
G = G / 255;
B = B / 255;
```

The RGB values are normalized to the range [0,1] by dividing each component by 255. This is a crucial step in color space conversions, ensuring correct computations.

Step 4: Apply Gamma Correction

The function defines an inner function:

```
ffunction gamma_corrected=apply_gamma_correction(channel)
        mask = channel > 0.04045;
        gamma_corrected = mask .* ((channel + 0.055) / 1.055) .^ 2.4 + (~mask .*
(channel / 12.92));
    endfunction
    R = apply_gamma_correction(R);
    G = apply_gamma_correction(G);
    B = apply_gamma_correction(B);
```

This corrects the gamma encoding of the RGB channels. If a pixel value is greater than 0.04045, it is adjusted using a power function; otherwise, it is scaled linearly.

Step 5: Convert to XYZ Color Space

Using the D65 illuminant transformation matrix:

```
X = R * 0.4124564 + G * 0.3575761 + B * 0.1804375;
Y = R * 0.2126729 + G * 0.7151522 + B * 0.0721750;
Z = R * 0.0193339 + G * 0.1191920 + B * 0.9503041;
```

These coefficients are derived from standard color science transformations.

Step 6: Adjust for D65 Reference Points

```
X = X * 200;
Y = Y * 250;
Z = Z * 200;
```

This scales the values to match the D65 reference white point.

Step 7: Clamping for Visualization

```
X = max(0, min(255, X));
Y = max(0, min(255, Y));
 Z = max(0, min(255, Z));
```

The XYZ values are clamped to the [0,255] range for proper visualization in an image processing context.

Testing the function

Load an RGB image

```
img = imread("image.jpg");
```

```
// Convert to XYZ color space
[X, Y, Z] = rgb2xyz(img);
```

Display the XYZ channels

```
imshow(X);
imshow(Y);
imshow(Z);
```

Error Handling

- If the input is not a valid RGB image, an error is raised: "Input must be an RGB image"
- If the number of arguments is incorrect, an error is raised: "Single RGB image expected"

For an RGB Image:

[X, Y, Z] = rgb2xyz(input_image);

- input_image: Sample image , image on which conversion is to be performed. This line calls the rgb2xyz function with the loaded image as the input. The function converts the image's RGB color channels into the XYZ color space. The function returns three separate matrices:
    - X: The X channel (representing the red-green component).
    - Y: The Y channel (representing luminance or brightness).
    - Z: The Z channel (representing the blue-yellow component).

Plotting individual X , Y and Z channels:

```
subplot(1,3,1); imshow(uint8(X)); title('X Channel’);
subplot(1,3,2); imshow(uint8(Y)); title('Y Channel');
subplot(1,3,3); imshow(uint8(Z)); title('Z Channel’);
```

- subplot() divides the figure into a 1x3 grid (1 row, 3 columns) and sets the current plot to the first position.
- imshow(uint8()); displays the channel (converted to uint8 for proper visualization), which represents the red-green component in the XYZ color space.
- title('__ Channel'); adds the title '__ Channel' to the first subplot.

Adding the individual channels to visualise image in XYZ colorspace:

```
xyz_img = cat(3, X, Y, Z);
```

- This function concatenates the X, Y, and Z channels along the third dimension (color channels) to create a single 3D array (xyz_img). This new image represents the image in the XYZ color space, with the three channels combined.

Creating a copy of the image:

`figure();`

This creates a new figure window for displaying the next plot (the XYZ image). It ensures that the following image will not overwrite the previous one.

Displaying the output:

`imshow(uint8(xyz_img));`

- This function displays the image (xyz_img) in the newly created figure. The image is first converted to uint8 for proper visualization. Since xyz_img contains the combined XYZ channels, it will display the image as a version in the XYZ color space, which may look different from the original RGB version.

Parameters:

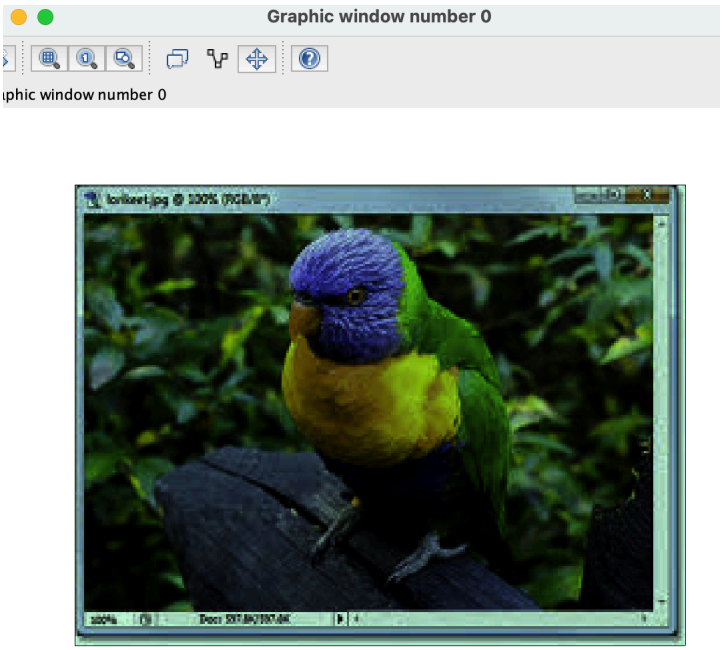
1. **input_image** :
    - Type: 3D array (height x width x 3)
    - Description: A color image represented by an RGB image matrix. The third dimension should be of size 3, corresponding to the red, green, and blue channels.
    - Valid Range: The RGB values should be within the range [0, 255] before processing.
2. **R** :
    - Type: 2D array (height x width)
    - Description: A matrix representing the red channel of the image. It should be in the range [0, 255].
    - Valid Range: [0, 255]
3. **G** :
    - Type: 2D array (height x width)
    - Description: A matrix representing the green channel of the image. It should be in the range [0, 255].
    - Valid Range: [0, 255]
4. **B** :
    - Type: 2D array (height x width)
    - Description: A matrix representing the blue channel of the image. It should be in the range [0, 255].
    - Valid Range: [0, 255]

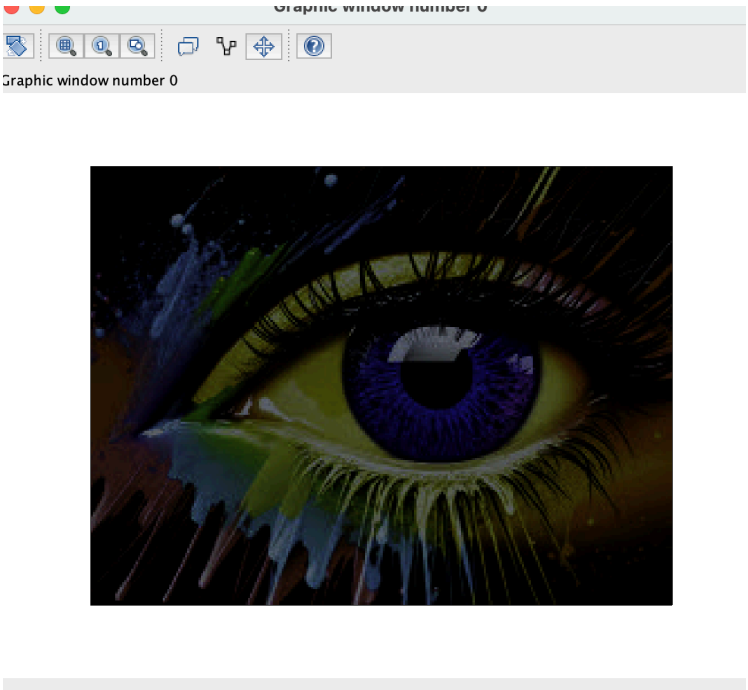
Conclusion

This rgb2xyz function efficiently converts RGB images into the XYZ color space while ensuring proper input validation, gamma correction, and scaling for visualization. The approach follows standard color science practices and maintains high numerical accuracy for image processing applications.

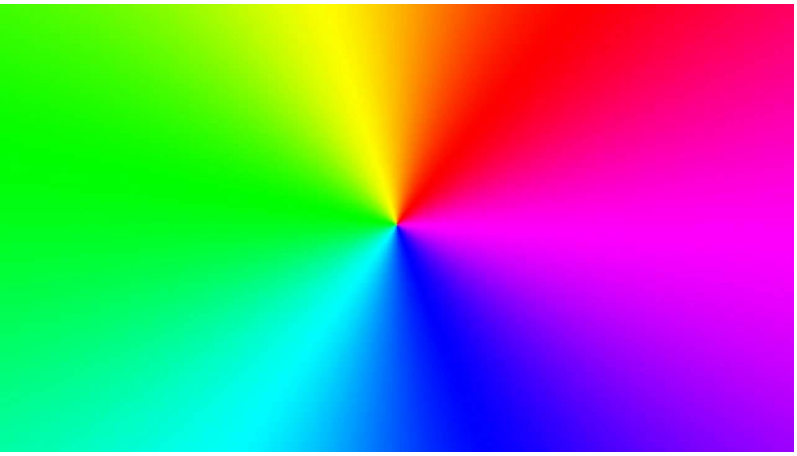
Input : RGB

Output : XYZ colorspace(brightness of output can be altered)

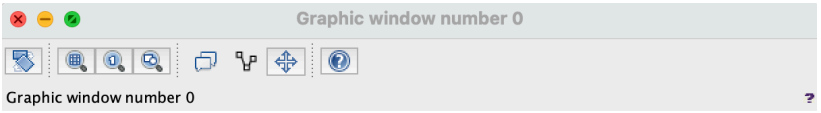


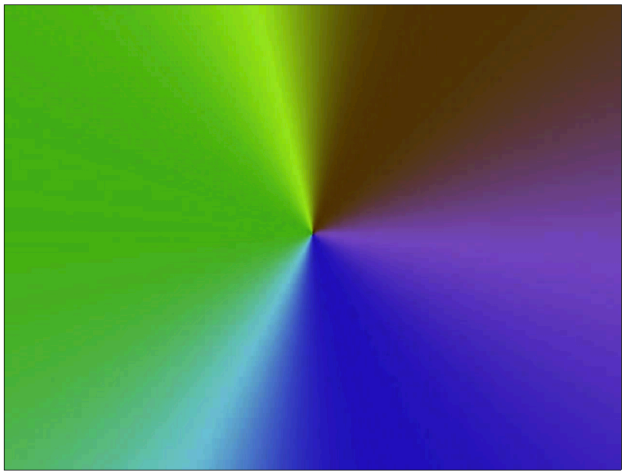


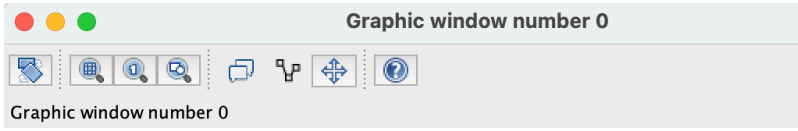
Input : RGB colorspace image

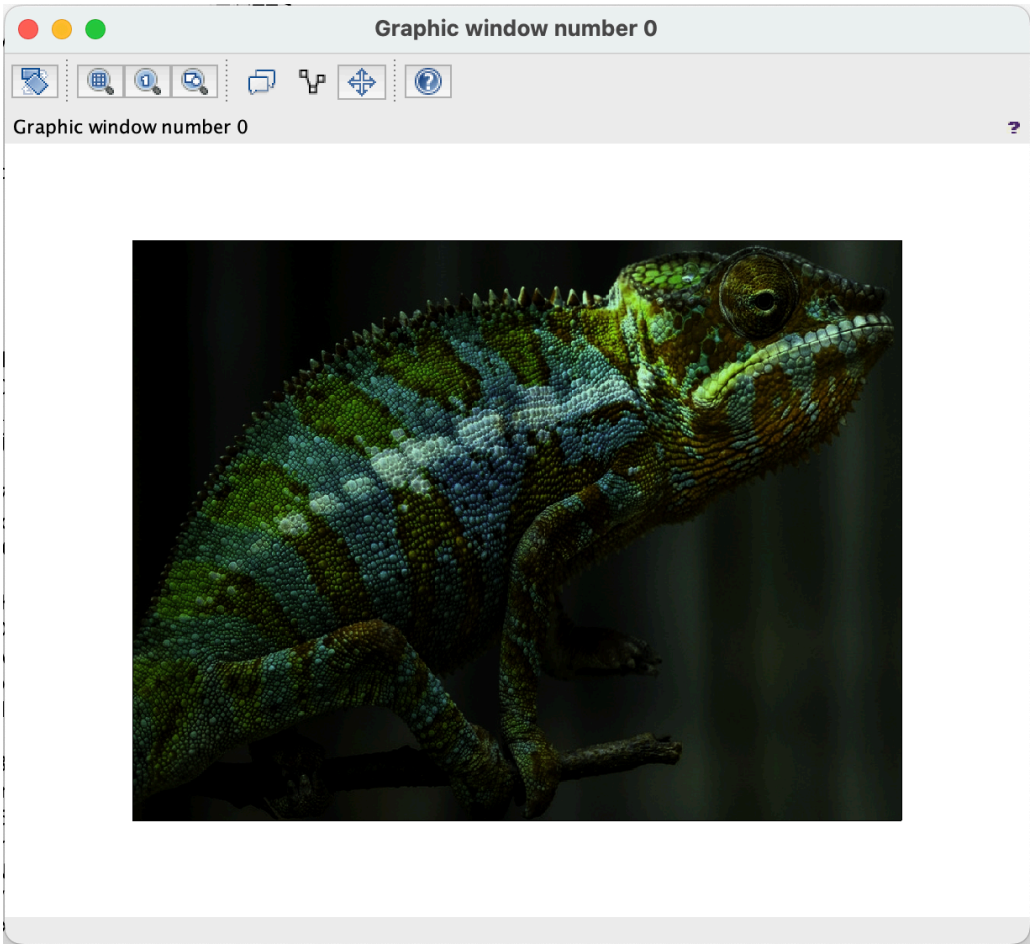


Output : XYZ colorspace conversion








Graphic window number 0
Graphic window number 0