Overview

The imapplymatrix function in Scilab takes an input multi-channel image (e.g., RGB) along with a transformation matrix and applies that transformation to correspondingly and independently each pixel in each channel of the image and outputs the resultant image. This is a common operation in image processing tasks such as color space conversion and application of colour transformation matrices.Specifically to each pixel on the input image X, it transforms it using matrix M followed by possible adding of vector C (typically used for color-shift or bias). Y is the transformed image returned.

Function Signature

Y = imapplymatrix(M, X, C)

M: Transformation matrix (q×p)

X: Input image (m × n × p), m is height, n is width, and p is number of color channels.

C: Optional bias or shift vector (size: q x 1) Notice if w is not passed by default w = 0_qx1

Parameters

1. M (Transformation Matrix)

- Type: Matrix (size q x p)
- It is used to convert the pixel values from one color space (or representation) to another. In this case we have our transformed output channels in the rows of the matrix and our color channels of the image as columns of the matrix.

2. X (Input Image)

- Type: 3D matrix (m x n x p)
- The input image is the 1st where:]
    - m =height = number of rows,
    - n is the number of columns(width),
    - p is the number of color channels (for RGB, we have p=3).
- The image should have p color channels that match the number of columns in the transformation matrix M.

3. C (Bias/Offset Vector)

- Type: Vector (size q x 1)

- The result of the matrix transformation is transformed or shifted by this optional vector. This is the most frequently used tool in color transformations when you want to change the color values after matrix transformation. In case it is not given, it is set to be a zero vector of the length q that is same as the number of channels in the output.

Outputs

Y (Transformed Image)

Type: 3D matrix (m x n x q)
Description: This is the output image that results from the matrix transformation M and optional bias C, where after applying M the image will have the q channels, with every channel being a transformation of the input channels according to M.

Explanation of Code

1. Input Validation

```
[[m, n, p] = size(X);
 [q, pM] = size(M);
if p <> pM then
    error("Matrix M  must have the same number of columns as no of color channels image has”);
end
```

- This section tests the dimensions of the input image X and transformation matrix M and ensure that the number of color channels p in X is equal to the number of columns in M (since each row in matrix M corresponds to a transformed channel of an image).
-

2. Checking C (Bias Vector)

```
if argn(2) < 3 then
        C = zeros(q, 1);
    end
    if length(C) <> q then
        error("C must be a vector of length that is equal to the number of output channels (rows in M).");
    end
```

- The effect of automobiles on communities has been recognized for a long time. Not only do they have a significant impact on pollution, but they also increase congestion and alienate people from their surroundings.

3. Initialize Output Image

```
Y = zeros(m, n, q);
```

- **Purpose**: This line initializes the output image Y with dimensions corresponding to the height m, width n, and the number of output channels q. Initially, it’s filled with zeros.

4. Transformation Loop

```
for i = 1:m
      for j = 1:n
         pixel = X(i, j, :);
         pixel = matrix(pixel, [p, 1]);
         Y(i, j, :) = M * pixel + C;
      end
   end
```

- This loop will iterate over every pixel in the input image X. For each pixel it: Extract pixel color values from all channels (RGB).
- Convert pixel values to column matrix with matrix(pixel, [p, 1])
- Apply transformation using matrix M and adding bias vector C
- Store the result in the corresponding pixels of the output image Y

Calling sequence

Step 1: Load an image

```
X = imread('image.png');  // Reads the input image
```

Step 2: Define a transformation matrix M

```
M = [0.299, 0.587, 0.114; 0.5957, -0.274, -0.321; 0.211, -0.522, 0.311];
```

Step 3: Apply the transformation

```
Y = imapplymatrix(M, X);
```

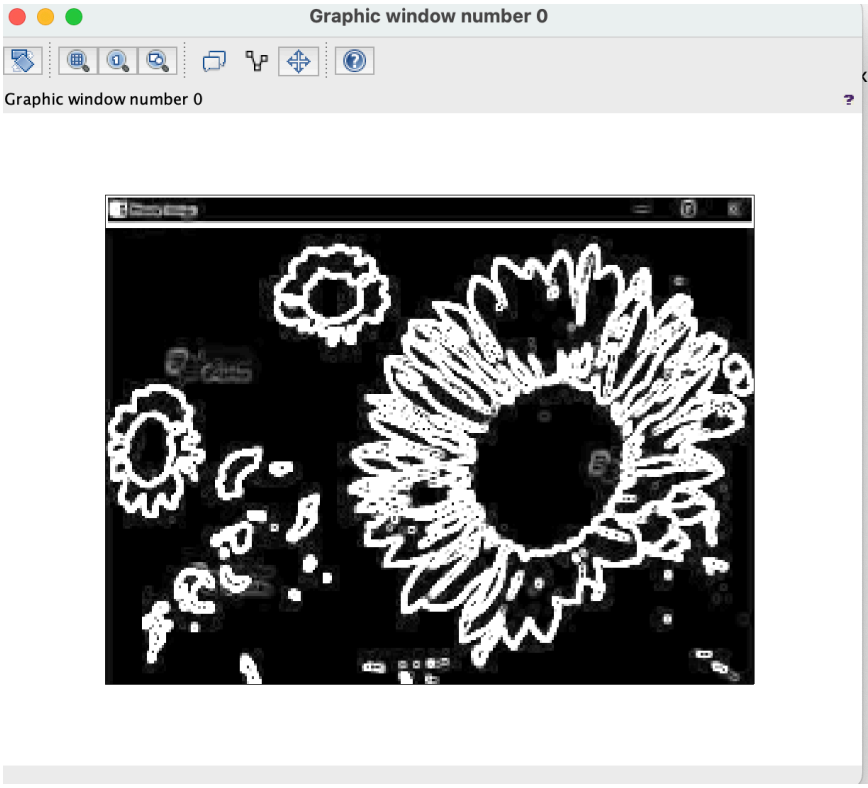
Step 4: Display the transformed image

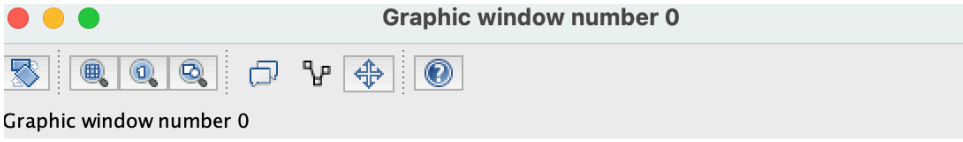
```
imshow(Y);  // Displays the transformed image
```

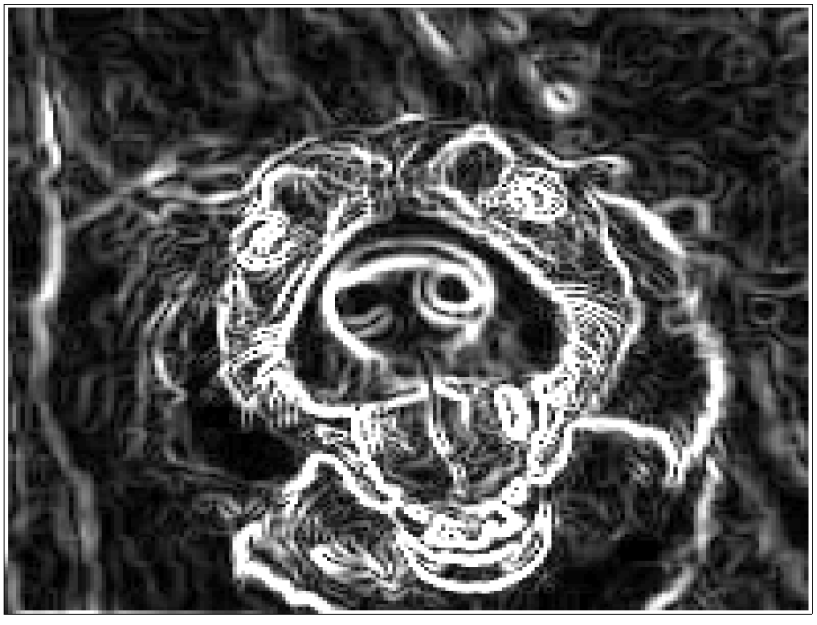
Conclusion

Imapplymatrix function It provides a flexible way to apply linear transformations to individual pixels of multichannel images. It can be used for a variety of image processing tasks, such as converting color space and applying various color filters. This should give the user a clear understanding of how to use and modify the functionality!

Input & output images:







Input images & output images: