Lab 01

Pixel Operations

Aaron Dinesh - 20332661

13 February 2024

Declaration

I hereby declare that this report is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university. I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`. I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `https://libguides.tcd.ie/academic-integrity/ready-steady-write`. I consent to the examiner retaining a copy of the report beyond the examining period, should they so wish. I agree that this thesis will not be publicly available, but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement. **Please consult with your supervisor on this last item before agreeing, and delete if you do not consent**

Signed: Aaron Dinesh　　　　Date: 13 February 2024

1 Image Loading and Display

1.1 Loading the image

I used the imread function to read the image. I then used the double function to convert the data type of the image to a double and finally normalized all the values to be in the range 0 - 1. The lab then required me to use the rgb2gray function to convert the RBG image to a grayscale image. This is shown below in Figure 1.

Figure 1: RBG and Grayscale Image

1.2 Adding 128 to the pixels

Here I had to add 128 to all the pixels in the RGB image, effectively brightening the image. However, since all the values in the image were normalized I had to add $\frac{128}{255}$ to all the pixels in the image. The output from this operation is shown below in Figure 2.

Figure 2: Adding 128 to the RGB Image

As can be seen from Figure 2, a lot of the pixels that were gray in the normal image turned white, or close to white, after the transformation. This is because colors that are gray or close to gray, have RGB values that are around (128, 128, 128). Adding 128 to these pixels means that their RGB values come close to (255, 255, 255) which corresponds to white.

1.3 Subtracting 128 from the pixels

In this exercise, I had to subtract 128 from all the pixels. This was performed in a similar manner to exercise 1.2. The result of this operation is shown in Figure 3. Pixels that have a value of gray or close to gray in the original image now appear black since they have their RBG values pushed to (0, 0, 0) after the transformation. This also has the effect of darkening the image when compared with the original.

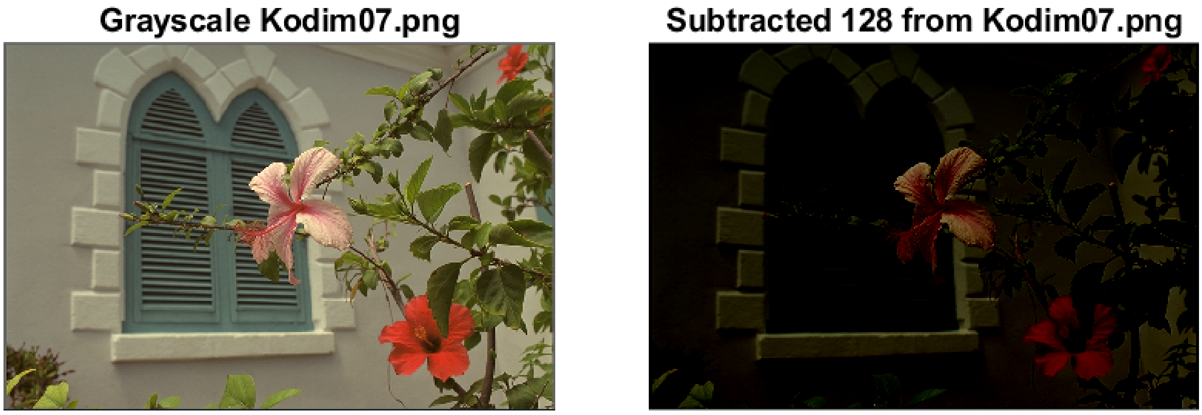

Figure 3: Subtracting 128 from the RGB Image

2 Histograms

2.1 Tennis.png RGB Histograms

The Tennis.png file was loaded using a similar method as before. Once the image was loaded, I separated the channels and passed each component to the histogram function in MATLAB which calculated the bins and produced the histogram. This is shown in Figure 4

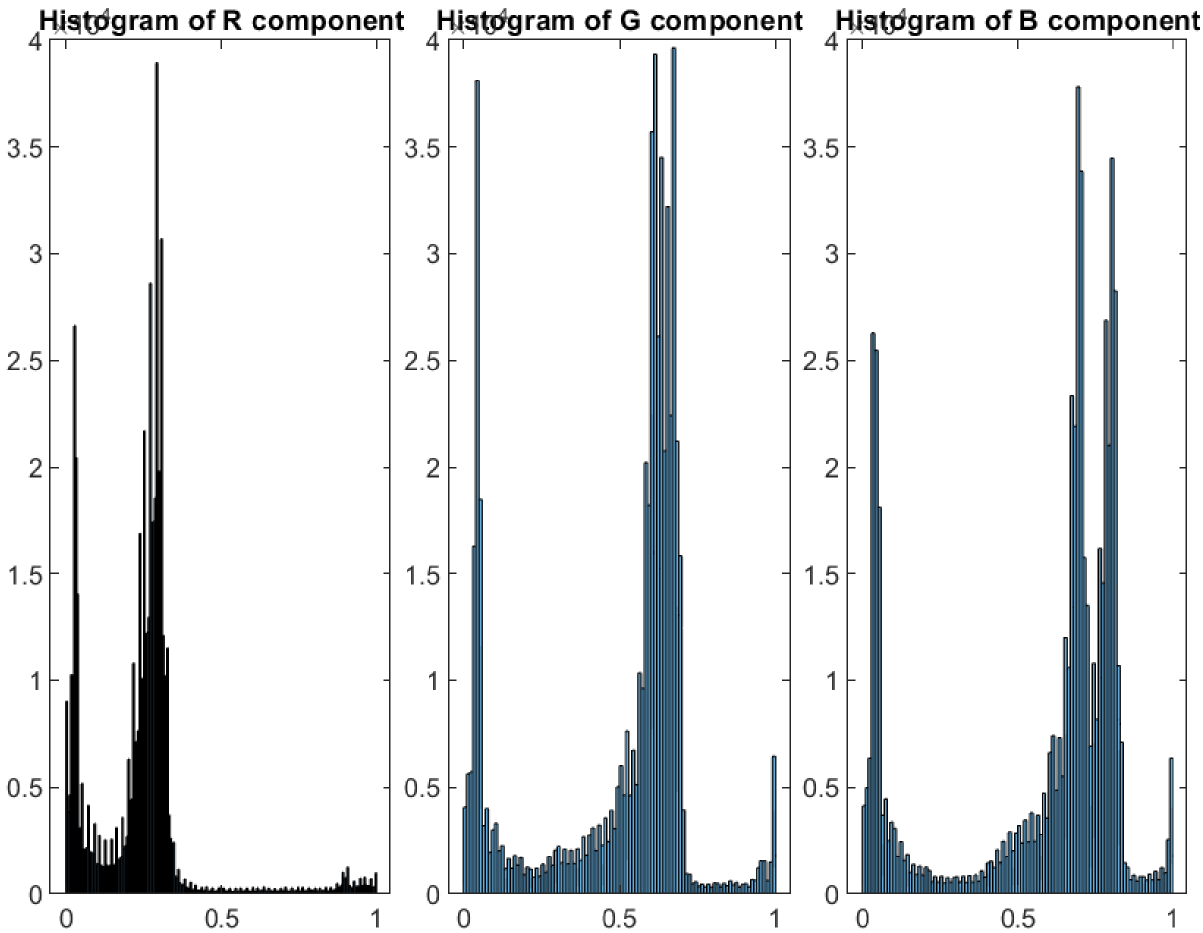

Figure 4: Histograms of the R, G and B components of Tennis.png

2.2 Conversion to YCbCr

Converting to the YCbCr space was as easy as calling the rgb2ycbcr function in MATLAB. The output of the function is shown in Figure 5.

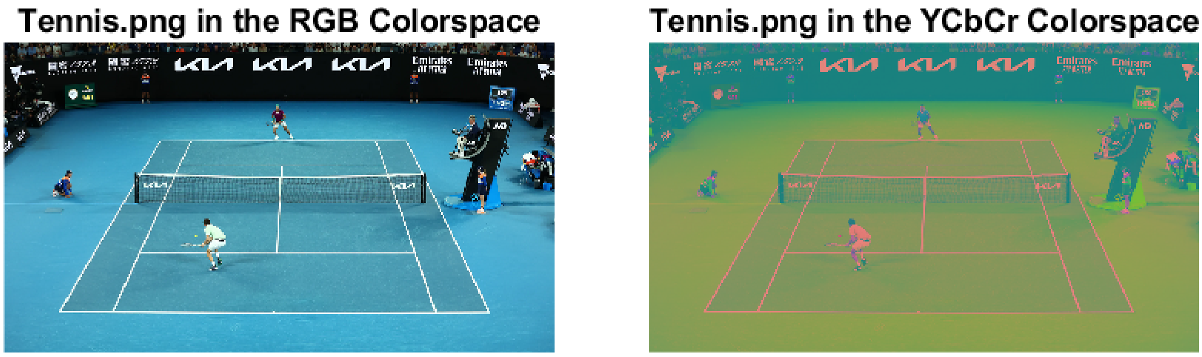

Figure 5: A comparison of Tennis.png in the RBG and YCbCr Color Space

2.3 Tennis.png YCbCr Histograms

Shown below are the histograms after the image was converted to the YCbCr Color space.

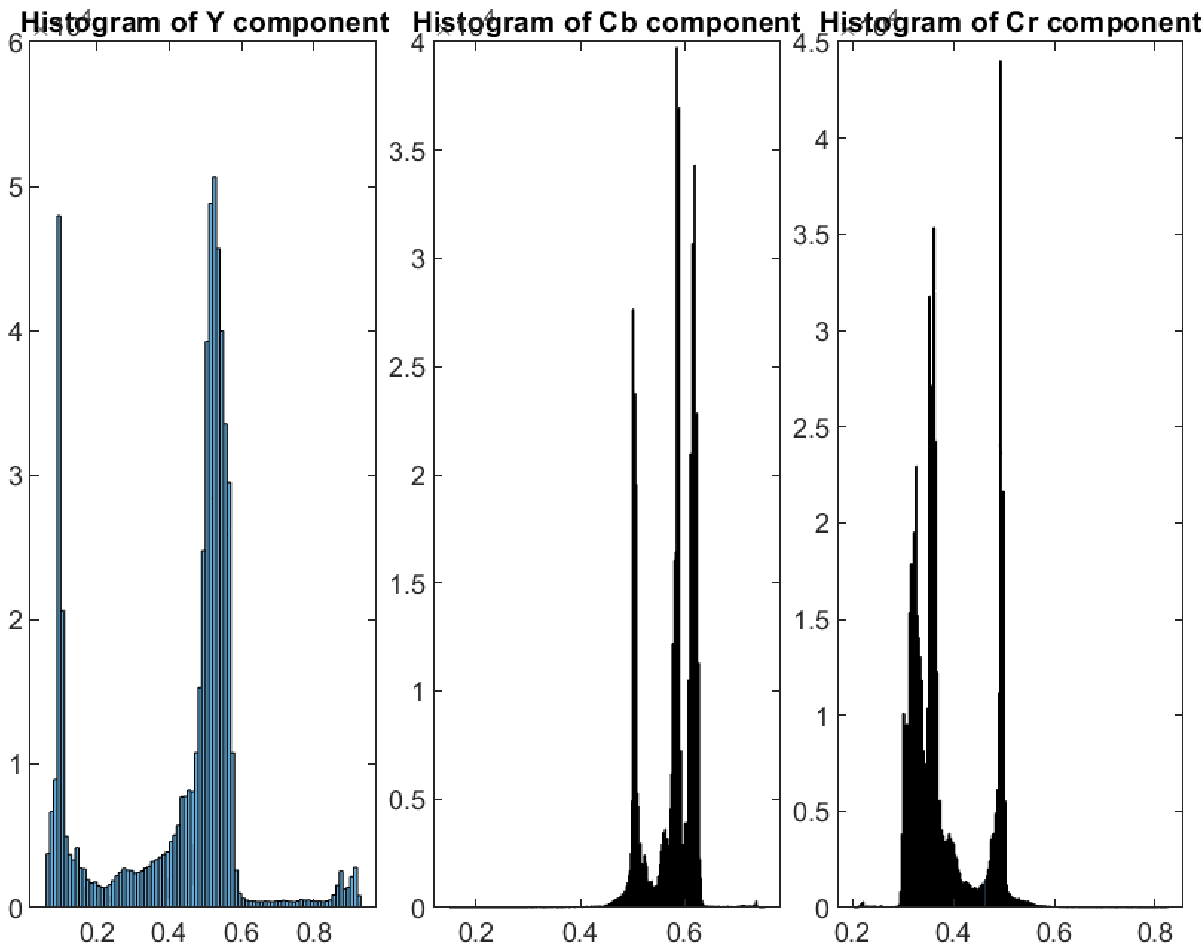

Figure 6: Histograms of the Y, Cb and Cr components of Tennis.png

2.4 Interpreting Histograms

Histograms allow us to see the distribution of the color components in our image. These are extensively used in color-grading applications for movies and TV shows. It allows graphics artists to hone in on the exact color profile that they envision. However, the information provided by the histograms can also be used to segment large objects in an image. The idea behind this is that large objects that are relatively homogeneous in color, will appear as spikes on the histograms. This can be seen in the YCbCr histograms in Figure 6. Large spikes can be seen on the Y component around 0.57, on the Cb component around 0.56, and on the Cr component around 0.48. We can then use these values as a threshold and create a mask that will only select the pixels that are above this threshold.

3 Segmentation

3.1 Segmenting the Court

Here I implemented the logic described in section 2.4. I used the RBG histograms to find suitable thresholding values for the R, G and B components. This created a mask for each color component. I then played around with various combinations of the component masks to settle on a final mask. This was the final combination of the color masks:

$$mask = redMask \wedge greenMask \vee blueMask$$

This mask was then multiplied element-wise with the original picture to segment the court, the results of which are shown below in Figure 7

Figure 7: Segmented Tennis Court

4 Color Balancing

4.1 Cumulative Histograms

In this section I had to calculate and plot the cumulative histograms for the R, G and B components for kodim23a.png and kodim23b.png (Figure 8). These histograms can be seen in Figure 9

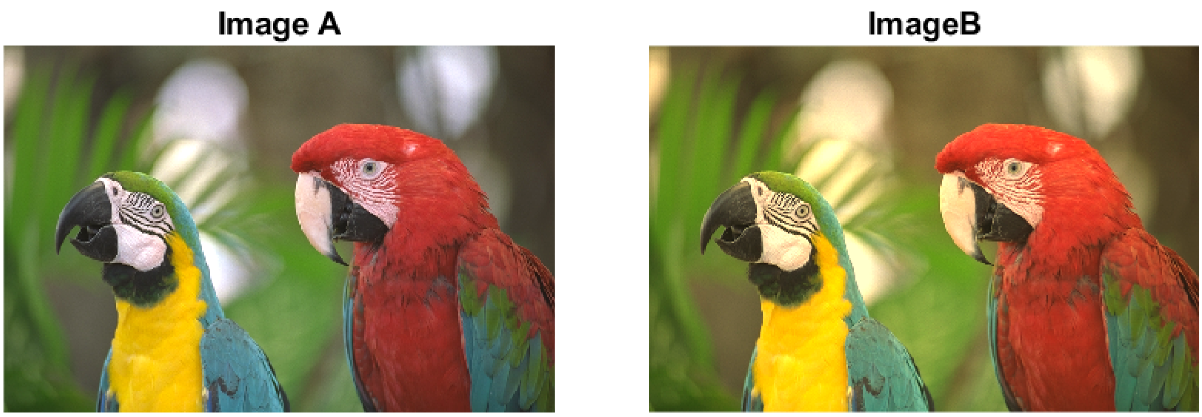

Figure 8: kodim23a.png and kodim23b.png

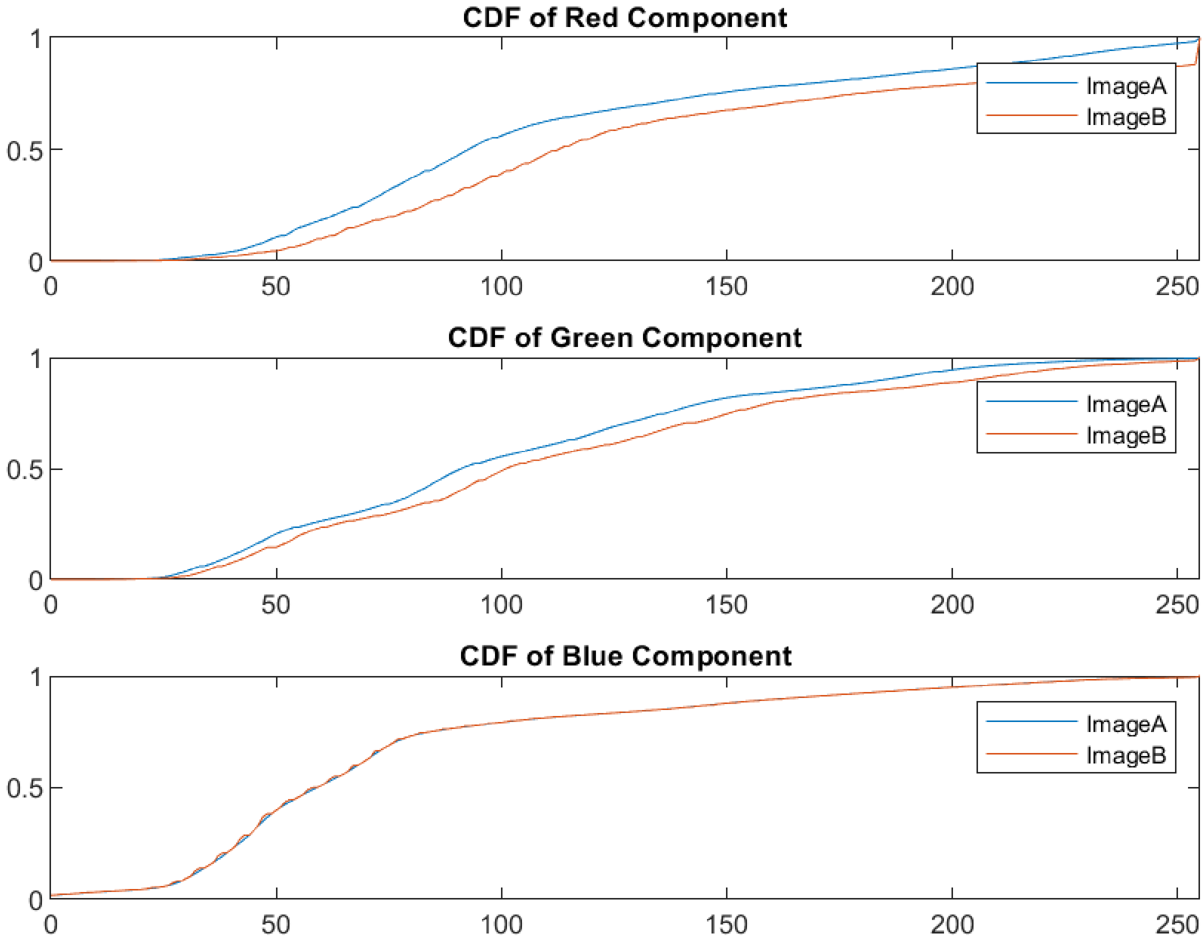

Figure 9: Cumulative Histograms for kodim23a and kodim23b

4.2 Color Mapping Kodim23b to Kodim23a

Using these histograms I calculated where the channel value of 150 in kodim23a got mapped to in kodim23b. These gave the values for $(R_a, G_a, B_a)^{\mathrm{T}}$ and $(R_b, G_b, B_b)^{\mathrm{T}}$. We can substitute these vectors into matrix 1, and solve the linear system to estimate the values for $(R_w, G_w, B_w)^{\mathrm{T}}$.

$$\begin{bmatrix} R_a \\ G_a \\ B_a \end{bmatrix} = \begin{bmatrix} \frac{255}{R_w} & 0 & 0 \\ 0 & \frac{255}{G_w} & 0 \\ 0 & 0 & \frac{255}{B_w} \end{bmatrix} \begin{bmatrix} R_b \\ G_b \\ B_b \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} 150 \\ 150 \\ 150 \end{bmatrix} = \begin{bmatrix} \frac{255}{R_w} & 0 & 0 \\ 0 & \frac{255}{G_w} & 0 \\ 0 & 0 & \frac{255}{B_w} \end{bmatrix} \begin{bmatrix} 182 \\ 167 \\ 150 \end{bmatrix} \tag{2}$$

$$150 = \frac{255 \times 182}{R_w}$$
$$150 = \frac{255 \times 167}{G_w}$$
$$150 = \frac{255 \times 150}{B_w}$$

$$
\begin{aligned}
R_w &= \frac{255 \times 182}{150} \\
B_w &= \frac{255 \times 167}{150} \\
G_w &= \frac{255 \times 150}{150}
\end{aligned}
$$

$$
\begin{bmatrix} R_a \\ G_a \\ B_a \end{bmatrix} = \begin{bmatrix} \frac{255\times150}{182\times255} & 0 & 0 \\ 0 & \frac{255\times150}{167\times255} & 0 \\ 0 & 0 & \frac{255\times150}{150\times255} \end{bmatrix} \begin{bmatrix} R_b \\ G_b \\ B_b \end{bmatrix} \tag{3}
$$

$$
\begin{bmatrix} R_a \\ G_a \\ B_a \end{bmatrix} = \begin{bmatrix} \frac{150}{182} & 0 & 0 \\ 0 & \frac{150}{167} & 0 \\ 0 & 0 & \frac{150}{150} \end{bmatrix} \begin{bmatrix} R_b \\ G_b \\ B_b \end{bmatrix} \tag{4}
$$

By performing this calculation on all the pixels of the image, we can color-match kodim23b to look more like Kodim23a. The result of performing this transformation can be seen in Figure 10, and the average absolute error between Kodim23a and the transformed Kodim23b can be seen in Figure 11

Figure 10: Tranformation of Kodim23b to color match Kodim23a

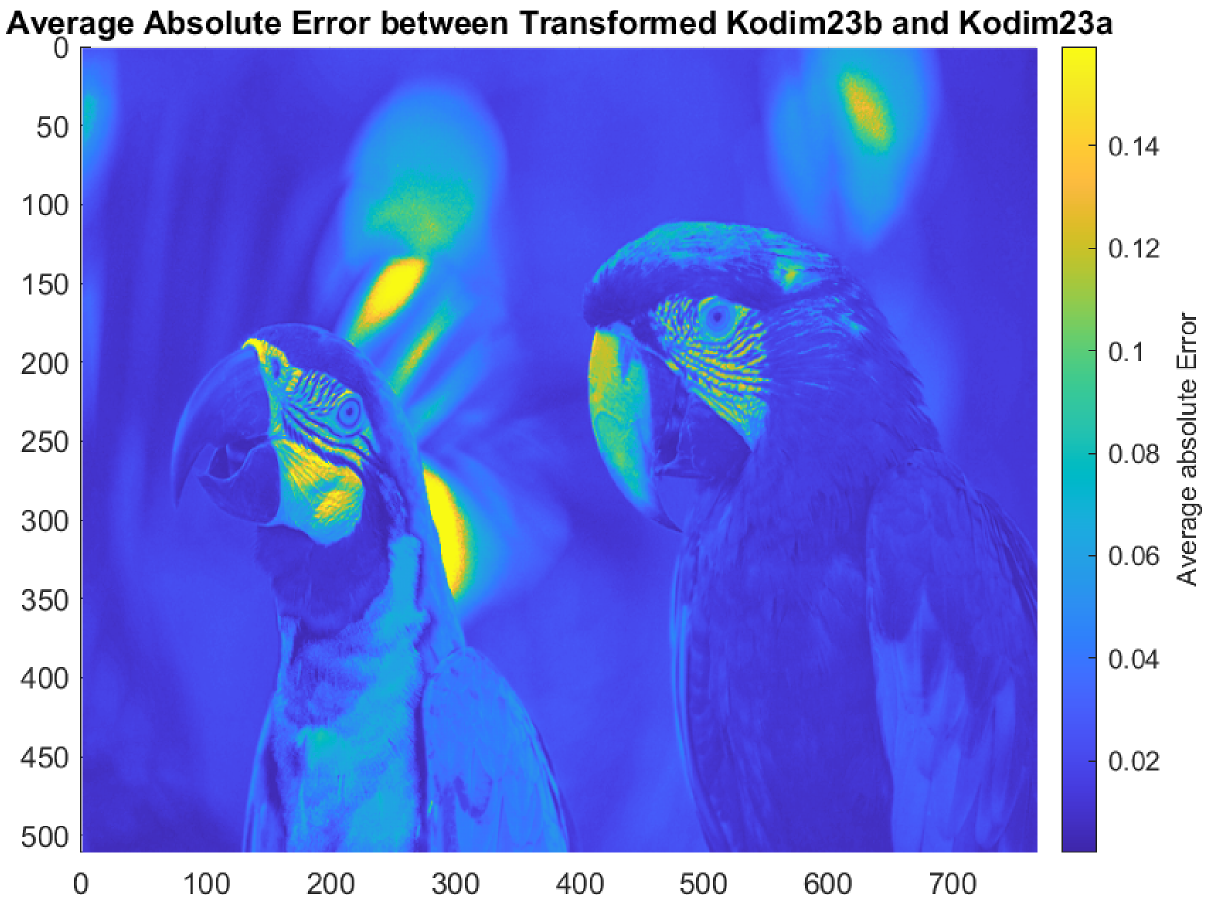

Figure 11: Average Absolute error between Kodim23a and the transformed Kodim23b