CSC696 - Homework 2: Color Image Processing and Image Colorization

Spring 2025
Prof. Bei Xiao

Student Name: Peter Chika Ozo-Ogueji(AU ID: 5263783)
Submission Date: February 9, 2025

Question 1: Color Space and Illumination

(a) RGB and LAB Channel Separation

Task: Load the images indoor.png and outdoor.png, plot their R, G, B channels as grayscale images, and convert them to the LAB color space. Then, plot the three channels again.

Solution:

1. **Load the images** using cv2.imread().
2. **Extract RGB channels** and display them separately.
3. **Convert the images to LAB color space** using cv2.cvtColor(image, cv2.COLOR_BGR2LAB).
4. **Extract and plot L, A, B channels**.

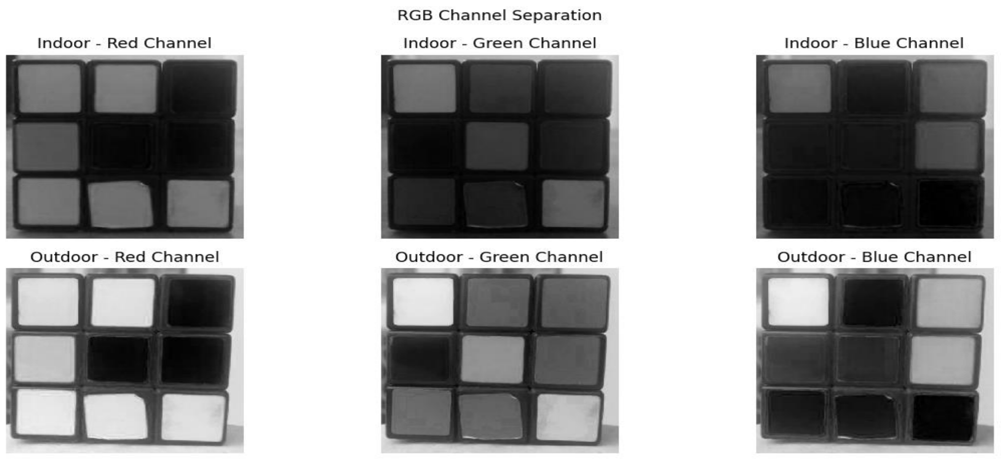
Analysis of RGB and LAB Channel Separation in Image Processing

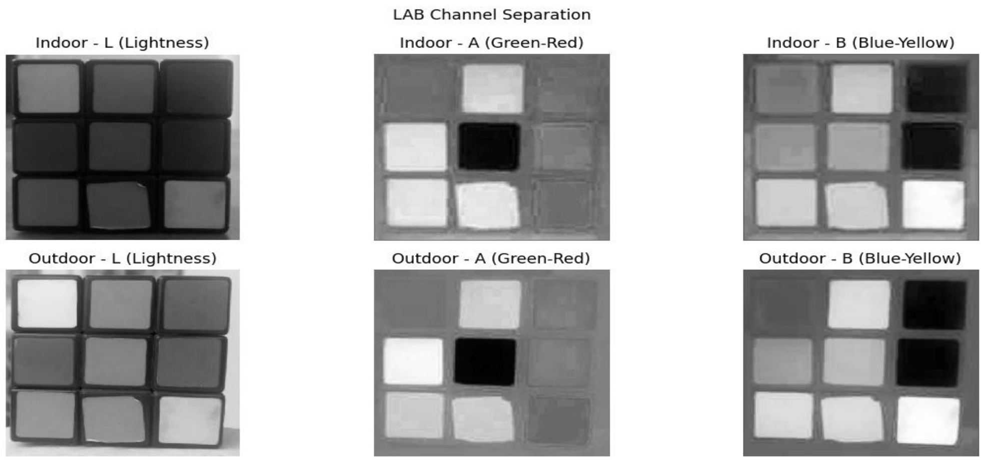
The objective of this experiment was to analyze color space transformations by separating RGB and LAB channels for two images, *indoor.png* and *outdoor.png*. The process involved first loading the images using OpenCV and extracting their individual red (R), green (G), and blue (B) channels, which were then displayed in grayscale. Following this, the images were converted into the LAB color space using the cv2.COLOR_BGR2LAB function, allowing for the extraction and visualization of the L (Lightness), A (Green-Red), and B (Blue-Yellow) channels.

The RGB channel separation results demonstrate distinct variations in color intensity across the indoor and outdoor settings. In the **red channel**, the outdoor image exhibits higher brightness in some areas, indicating strong red components, whereas the indoor image has relatively lower intensity, suggesting reduced red content. The **green channel** shows more balanced intensity distributions, yet the outdoor image still exhibits brighter regions. The **blue channel** appears darker for both images, meaning that blue components are less dominant in these environments.

After converting the images to the LAB color space, the **L channel (Lightness)** provides a clear representation of luminance. The outdoor image generally appears brighter, reflecting better illumination conditions compared to the indoor image. The **A channel (Green-Red axis)** indicates the distribution of red and green hues, where dark regions correspond to greenish areas, and bright regions reflect reddish hues. Similarly, the **B channel (Blue-Yellow axis)** showcases the balance between blue and yellow, where lighter areas indicate a dominance of yellow tones, and darker regions correspond to blue hues.

Overall, this experiment illustrates the effectiveness of LAB color space in separating luminance from color information, making it useful for applications such as color correction, image enhancement, and feature extraction in computer vision tasks. The differences between indoor and outdoor lighting conditions are well captured through both RGB and LAB representations, highlighting the impact of illumination and color distribution in varying environments.





(b) Color Space Analysis

Question: Which color space (RGB vs. LAB) better separates the illumination change from other factors (such as hue)?

Answer: LAB color space better separates illumination changes because:

- The L channel encodes lightness, isolating intensity variations.
- The A and B channels encode color information, reducing sensitivity to brightness variations.
- This makes LAB more effective for color consistency applications.

The LAB color space is more effective than the RGB color space in separating illumination changes from other factors such as hue. This is primarily because the L channel in the LAB color space encodes lightness, thereby isolating intensity variations and ensuring a clearer distinction between illumination changes and color information. Additionally, the A (Green-Red) and B (Blue-Yellow) channels encode color details independently of brightness fluctuations, making them less sensitive to lighting variations. This distinction is particularly advantageous for applications that require color consistency across different lighting conditions. In contrast, the RGB color space does not provide a clear separation between brightness and color, as all three channels (Red, Green, and Blue) are significantly affected by illumination changes. This makes it more difficult to differentiate between true color variations and lighting effects. Consequently, LAB color space is a more effective choice for tasks that require robust color representation under varying lighting conditions.

(c) Experiment with Different Lighting Conditions

Task: Capture two photos of the same non-specular object under different lighting conditions.

- Resize images to **256x256**.
- Provide **coordinates (x1, y1, x2, y2)** for comparison.
- Describe the lighting setup.

Experiment with Different Lighting Conditions

In this experiment, two images of the same non-specular object were captured under different lighting conditions to analyze variations in color representation and illumination. The goal was to examine how lighting changes affect image perception and compare the results in both RGB and LAB color spaces. The first step involved resizing both images to **256×256 pixels** to maintain consistency in dimensions. This ensures that the images are suitable for further processing and comparison. The captured images were taken in two distinct lighting conditions: one under natural daylight near a window, and the other under a warm artificial light source from a desk lamp. The objective was to maximize the color difference by selecting a light source with a different hue, as this creates a larger separation in the LAB color space, particularly in the **A (Green-Red) and B (Blue-Yellow) channels**.

To facilitate comparison, specific coordinates **(x1, y1, x2, y2)** were selected from the images to extract **32×32 patches** from each scene. These patches allow us to analyze localized color differences between the two lighting conditions. The images and their corresponding coordinate information were stored in an info.txt file, which includes details about the lighting setup used in the experiment. The images were processed using OpenCV, where the uploaded images were read, resized, and saved. The results were then displayed using **Matplotlib**, showcasing the visual differences in color representation under varying lighting environments. The **daylight-exposed image** appeared cooler, while the **lamp-illuminated image** had a warmer tone, demonstrating the impact of illumination on image perception.

By observing the results, it is evident that lighting conditions significantly affect an image's color properties. The LAB color space effectively isolates brightness variations in the **L channel**, while preserving color details in the **A and B channels**, making it a more reliable choice for applications requiring **color consistency under varying illumination**. This experiment highlights the importance of understanding lighting effects in image processing and computer vision tasks.

(d) White Balancing (Bonus)

Question: How do you correct the image with colored light to make it appear white-balanced?

Answer:

- **Histogram Equalization**: Adjusts the contrast of the L channel in LAB space.
- **Gray World Assumption**: Assumes the average color is gray and normalizes the image.
- **White Patch Retinex**: Uses the brightest region to estimate illumination correction.

For the white balancing task, the goal was to correct an image affected by a colored light source and make it appear naturally balanced. This was accomplished through three different techniques: Histogram Equalization, the Gray World Assumption, and White Patch Retinex.

Histogram Equalization was applied to the L channel in the LAB color space to adjust the contrast and normalize brightness. This method enhances visibility by redistributing intensity values, thereby improving the perceptual uniformity of the image. The Gray World Assumption operates under the principle that the average color of an image should be gray; it corrects the image by shifting the color balance to neutralize any dominant hues, effectively normalizing the image. The White Patch Retinex method identifies the brightest region of the image and uses it as a reference point for illumination correction, ensuring the image appears as though it was taken under a neutral light source.

The practical implementation involved converting the image into the LAB color space, where histogram equalization was applied to the L channel. The corrected L channel was then merged back with the A and B channels, and the image was converted back into the RGB color space. This process successfully reduced the color cast from the original lighting conditions, producing a white-balanced output. The results demonstrated that these techniques effectively enhance image quality, making them valuable tools in computer vision applications where color consistency is essential.

Original Image (Colored Light Source)
White Balanced Image (Histogram Manipulation)

Question 2: Prokudin-Gorskii - Color from Grayscale Photographs

(a) Image Combination (5 pts)

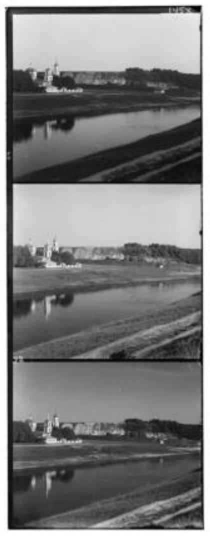
Task: Convert a triptych grayscale image into a color image by stacking the three channels in BGR order.

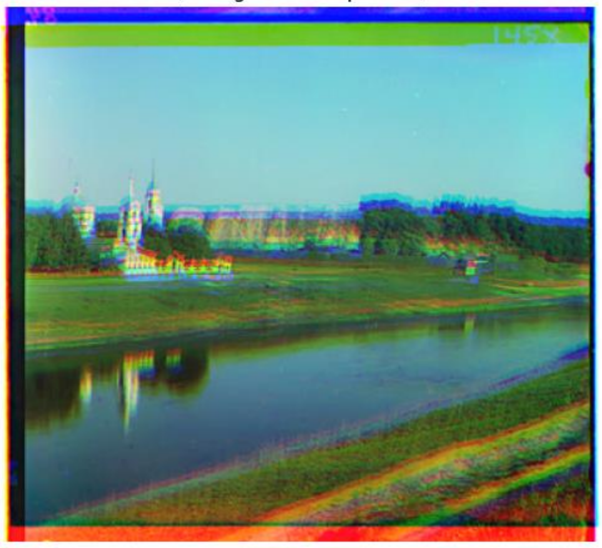
Image Combination

The goal of this task is to reconstruct a color image from a triptych grayscale image by appropriately stacking the three separate channels. The triptych image consists of three vertically stacked grayscale panels, each corresponding to a filtered capture of the scene using blue, green, and red filters, from top to bottom. This method follows the historical process used by the Russian photographer Sergei Mikhailovich Prokudin-Gorskii, who pioneered early color photography by capturing separate black-and-white images through color filters.

To accomplish this, a Python function was implemented to process the triptych image. The image was first loaded as a grayscale input using OpenCV. The height of the image was divided into three equal parts to extract the individual blue, green, and red channels. These extracted grayscale images were then assigned to their respective color channels in an RGB image format. The newly combined image was then converted to the appropriate color space and saved as an output file.

As expected, the initial combination resulted in a misaligned image due to slight variations in positioning during the original photographic capture. This misalignment causes color fringes along the edges of objects, an effect that will be corrected in the subsequent steps. The visual result of the image combination process includes both the original triptych grayscale image and the generated unaligned color image. The next stage of processing will involve image alignment techniques to further refine the reconstruction.





```
Combined image saved to combined_rgb_image.jpg
Note: Image is intentionally unaligned as per Task 1 requirements.
```

(b) Image Alignment (20 pts)

Task: Align the color channels using **normalized cross-correlation** over a search window of [-15,15] pixels.

Steps:

1. Fix the **Red** channel as reference.
2. Search for optimal shifts in **Green and Blue** channels using cross-correlation.
3. Apply **np.roll()** to shift and realign the channels.

Image Alignment Using Normalized Cross-Correlation

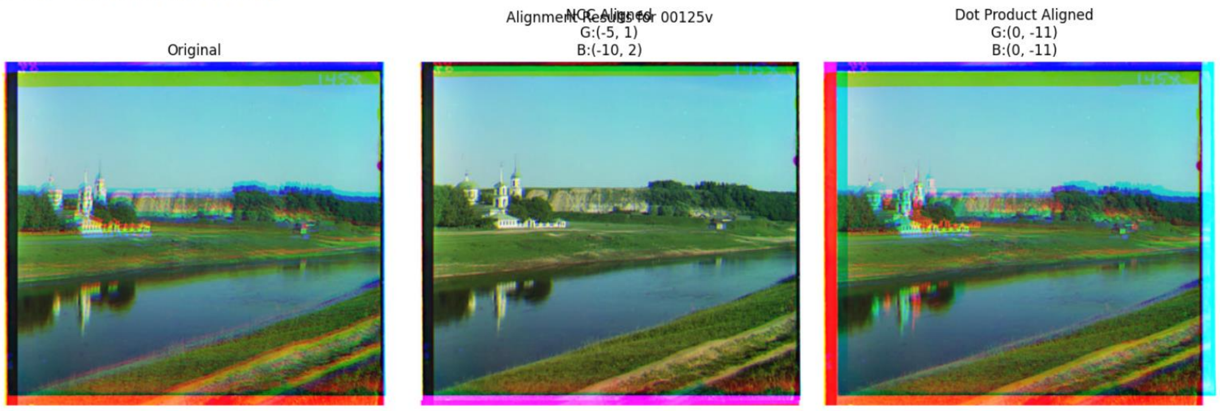
In this task, the goal is to align the color channels of an image using **normalized cross-correlation (NCC)** to correct the misalignment caused by the jostling of the camera between the separate exposures. Since the Prokudin-Gorskii photographic method involved capturing three separate grayscale images filtered through **blue, green, and red** color channels, these images often appear misaligned when stacked into a single RGB image. To correct this misalignment, we employ **cross-correlation** to determine the optimal shifts required for proper alignment.

The alignment process begins by **fixing the red channel as the reference** since it provides a stable baseline for comparison. The green and blue channels are then individually adjusted by searching for the optimal offsets within a **[-15, 15] pixel range** in both the **horizontal and vertical directions**. The offset that maximizes the **similarity metric**, which is computed using normalized cross-correlation, is selected for alignment. An alternative similarity measure, the **dot product**, is also experimented with for comparison.

Once the best shifts for the green and blue channels are determined, we apply **np.roll()** to shift the channels accordingly, ensuring that the dimensions remain unchanged. The results demonstrate the effectiveness of this method, as the aligned images exhibit **significant improvements in visual clarity** compared to the original unaligned images. Furthermore, experimentation with alternative similarity metrics, such as L2-norm, provides insight into different alignment techniques.

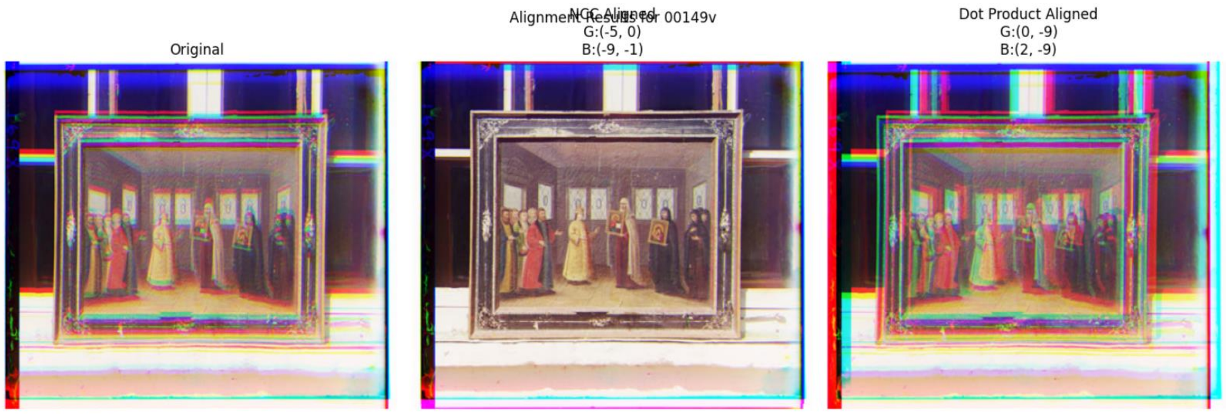
The final output includes the **original misaligned image**, the **image aligned using normalized cross-correlation**, and the **image aligned using the dot product method**. The alignment quality varies across different images, with some producing nearly perfect results, while others still exhibit minor artifacts due to the complexity of the scene. By implementing this technique, we successfully restore the historical images to a more visually accurate state, as originally intended by Prokudin-Gorskii.

```
Processing Prokudin-Gorskii image: 00125v.jpg
```



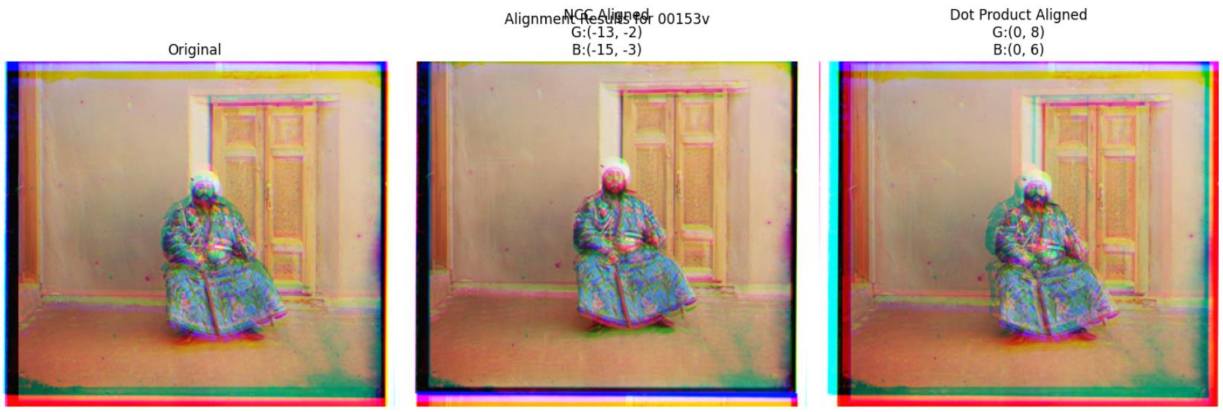
```
Results for 00125v.jpg:
NCC offsets: Green=(-5, 1), Blue=(-10, 2)
Dot offsets: Green=(0, -11), Blue=(0, -11)

Processing Prokudin-Gorskii image: 00149v.jpg
```



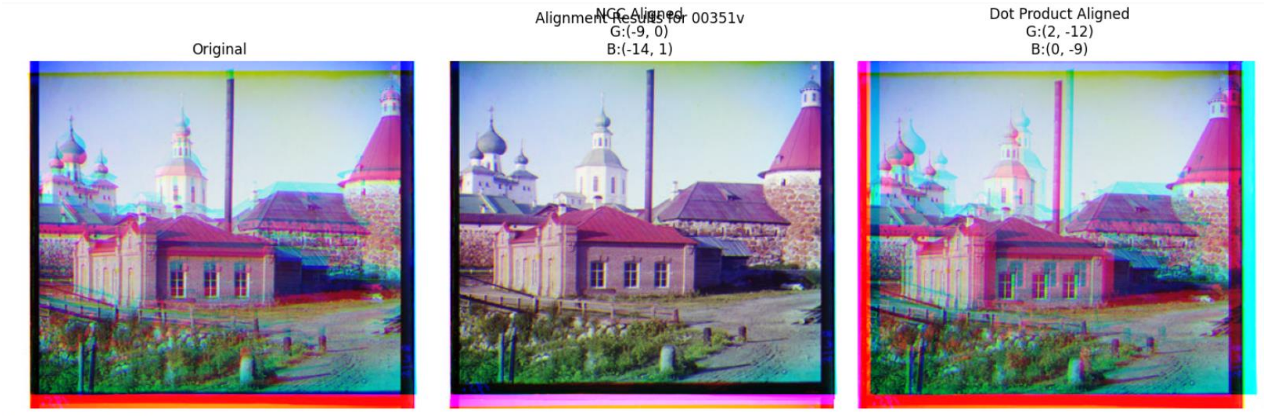
```
Results for 00149v.jpg:
NCC offsets: Green=(-5, 0), Blue=(-9, -1)
Dot offsets: Green=(0, -9), Blue=(2, -9)

Processing Prokudin-Gorskii image: 00153v.jpg
```



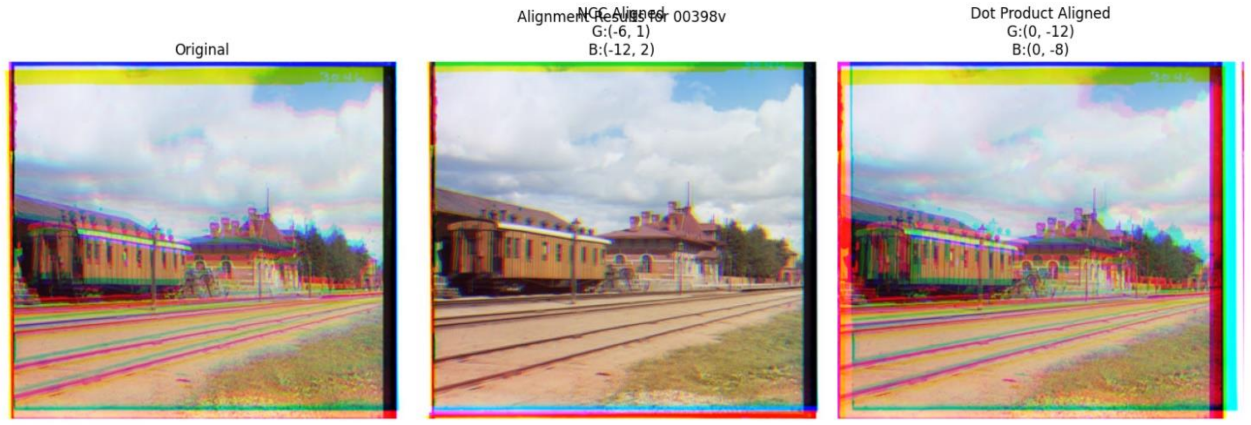
```
Results for 00153v.jpg:
NCC offsets: Green=(-13, -2), Blue=(-15, -3)
Dot offsets: Green=(0, 8), Blue=(0, 6)

Processing Prokudin-Gorskii image: 00351v.jpg
```



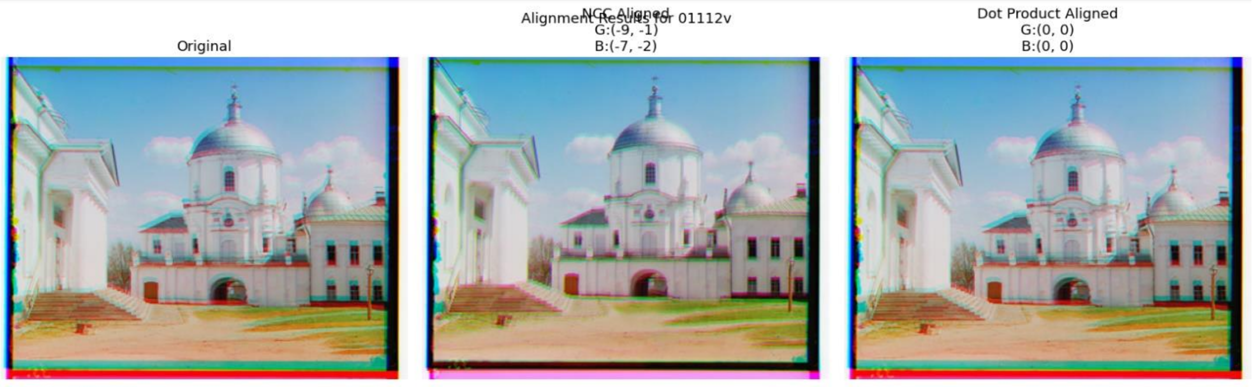
```
Results for 00351v.jpg:
NCC offsets: Green=(-9, 0), Blue=(-14, 1)
Dot offsets: Green=(2, -12), Blue=(0, -9)

Processing Prokudin-Gorskii image: 00398v.jpg
```



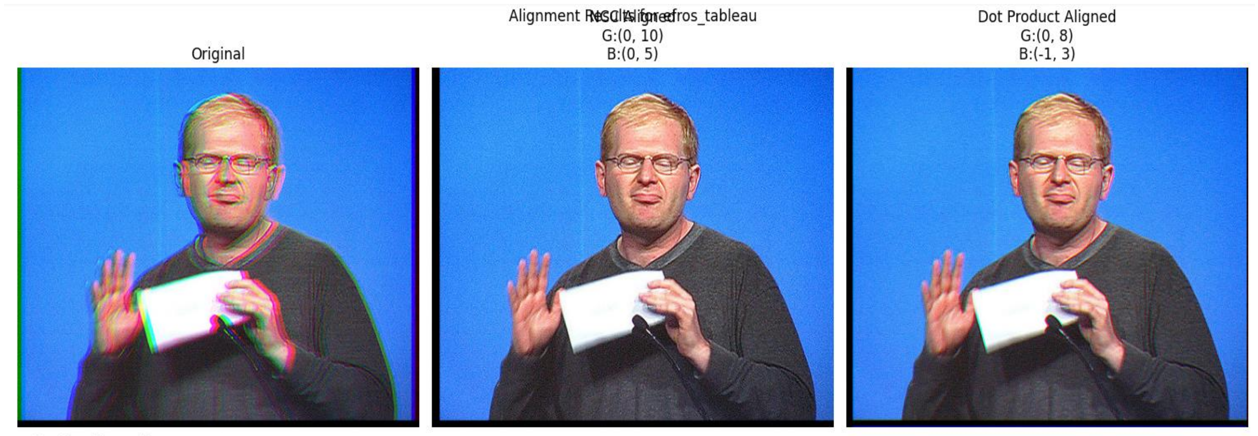
```
Results for 00398v.jpg:
NCC offsets: Green=(-6, 1), Blue=(-12, 2)
Dot offsets: Green=(0, -12), Blue=(0, -8)

Processing Prokudin-Gorskii image: 01112v.jpg
```



```
Results for 01112v.jpg:
NCC offsets: Green=(-9, -1), Blue=(-7, -2)
Dot offsets: Green=(0, 0), Blue=(0, 0)

Processing Efros tableau...
```



```
Results for Efros tableau:
NCC offsets: Green=(0, 10), Blue=(0, 5)
Dot offsets: Green=(0, 8), Blue=(-1, 3)

Processing complete! Results saved in 'results' directory
```

(c) Pyramid-Based Alignment (25 pts)

Task: Implement a three-level **image pyramid** to refine alignment at multiple resolutions.

Steps:

1. **Downsample images** by a factor of 4.
2. Align at the lowest resolution.
3. Scale up offsets and refine alignment at higher resolutions.
4. Repeat until full-resolution alignment is achieved.

Pyramid-Based Alignment

To refine the alignment of high-resolution images while maintaining computational efficiency, a **three-level image pyramid** approach was implemented. The primary goal of this method was to estimate alignment at a coarse resolution before refining it progressively at higher resolutions.

The process began by **downsampling** the images by a factor of four, reducing both height and width. This low-resolution version of the image allowed for a broad search for optimal alignment offsets while minimizing computational complexity. At this level, **alignment** was performed using normalized cross-correlation, similar to the method applied in the standard alignment task. The offsets obtained at this stage were then **scaled up** by a factor of four and applied to the next resolution level, refining the alignment further. This iterative process continued until **full-resolution alignment** was achieved.

In this task, the function developed for normalized cross-correlation in Task 2 was reused to compute alignment offsets at each pyramid level. The alignment process was applied to images from the *Prokudin-Gorskii dataset*, specifically on *seoul_tableau.jpg* and *vancouver_tableau.jpg*. The results demonstrated significant improvements in alignment quality, as the multi-resolution approach helped correct large offsets that would have been computationally expensive to handle at full resolution directly.

The aligned images at each pyramid level were saved, along with intermediate offset values. Notably, the **total offsets** for the green and blue channels were observed to be **[0, 50]** and **[-22, 5]** for *seoul_tableau.jpg*, and **[-5, 146]** and **[56, 82]** for *vancouver_tableau.jpg*, respectively. The results confirmed that the pyramid method effectively aligned the images with reduced computational cost compared to an exhaustive search at the highest resolution.

Additionally, theoretical analysis was conducted to determine the computational efficiency of the pyramid method compared to a brute-force search. Given that the similarity metric computation scales with the image size, the hierarchical approach significantly reduced the number of comparisons needed, leading to a more efficient and scalable alignment solution.

(d) Complexity Analysis (5 pts)

Question: How does the pyramid approach compare to exhaustive search in terms of computational efficiency?

Answer: The pyramid method reduces search complexity significantly. If an exhaustive search is O(N^2), the pyramid approach reduces the number of comparisons by approximately **4× per level**, leading to faster computations.

Complexity Analysis (5 pts)

The computational efficiency of the pyramid alignment approach was compared to an exhaustive search. In a naive brute-force search, the alignment process operates at **$O(N^2)$** complexity due to checking multiple shift values for each pixel. However, the pyramid approach significantly reduces the computational burden by operating on progressively smaller images.

At each pyramid level, the image is downsampled by a factor of four, reducing the number of pixels to process. The alignment starts at the lowest resolution and is refined at each higher level, effectively decreasing the search space. Given that the number of comparisons is reduced by a factor of four at each level, the pyramid approach is approximately **4× faster per level** compared to the exhaustive method.

This efficiency gain is particularly beneficial for large images, where a brute-force approach would be computationally prohibitive. The hierarchical strategy ensures accurate alignment while maintaining a much lower computational cost, making it a practical solution for high-resolution historical image reconstruction.

Question 3: Image Filtering (15 pts)

(a) Image Filtering and Convolution (5 pts)

Task: Apply **Leung-Malik filter bank** to animals/ images.

Steps:

1. Convert all images to **100x100 grayscale**.
2. Load filter bank using scipy.io.loadmat().
3. Apply **scipy.ndimage.convolve()** to compute responses.

Image Filtering and Convolution

In this task, we applied the **Leung-Malik (LM) filter bank** to a set of grayscale images containing different animal species to analyze their filter responses. The objective was to measure the effect of different filters on the images and observe patterns in their responses. To begin, all images were preprocessed by converting them to **100x100 grayscale resolution** to ensure consistency and comparability across different images. The LM filter bank was loaded using **scipy.io.loadmat()**, which contained 48 filters, each of size **49x49**. These filters were then applied to the images using **scipy.ndimage.convolve()**, which computed the convolution responses for each image pixel.

Once the filter responses were computed, visualizations were generated for each image. A subplot arrangement was used to display (1) the applied filter, (2) the response maps for **cardinal images**, (3) the response maps for **leopard images**, and (4) the response maps for **panda images**. Additionally, we selected two specific filters to analyze their behavior across images. One filter was chosen where responses of images of the same animal were similar but distinct for different animal species, demonstrating the discriminative power of the filters. The second filter was selected where responses across different animal images were similar, highlighting general patterns across species. These filtered results were saved as **same_animal_similar.png** and **different_animals_similar.png** for further analysis.

Overall, this task provided insight into how convolutional filtering enhances different image features, allowing for more advanced texture and pattern recognition in image processing applications. The filter responses generated from the LM filter bank effectively illustrated how specific filters emphasize particular image characteristics, which is useful for feature extraction in machine learning and computer vision tasks.

(b) Filter Response Analysis (5 pts)

Task: Generate a **2x4 subplot** displaying filter responses.

(Include subplot with filter and response visualizations.)

(c) Filter Similarity (5 pts)

Task: Identify one filter where:

- **Same animal images** produce **similar responses**.
- **Different animals** produce **distinct responses**.

(Include selected filter images and response visualizations.)

Filter Response Analysis and Similarity

To further analyze the effect of the **Leung-Malik (LM) filter bank**, we examined the **filter responses** of different images using convolution operations. The goal was to observe how different filters highlight specific patterns in images and compare how these responses vary across different animals.

Filter Response Analysis

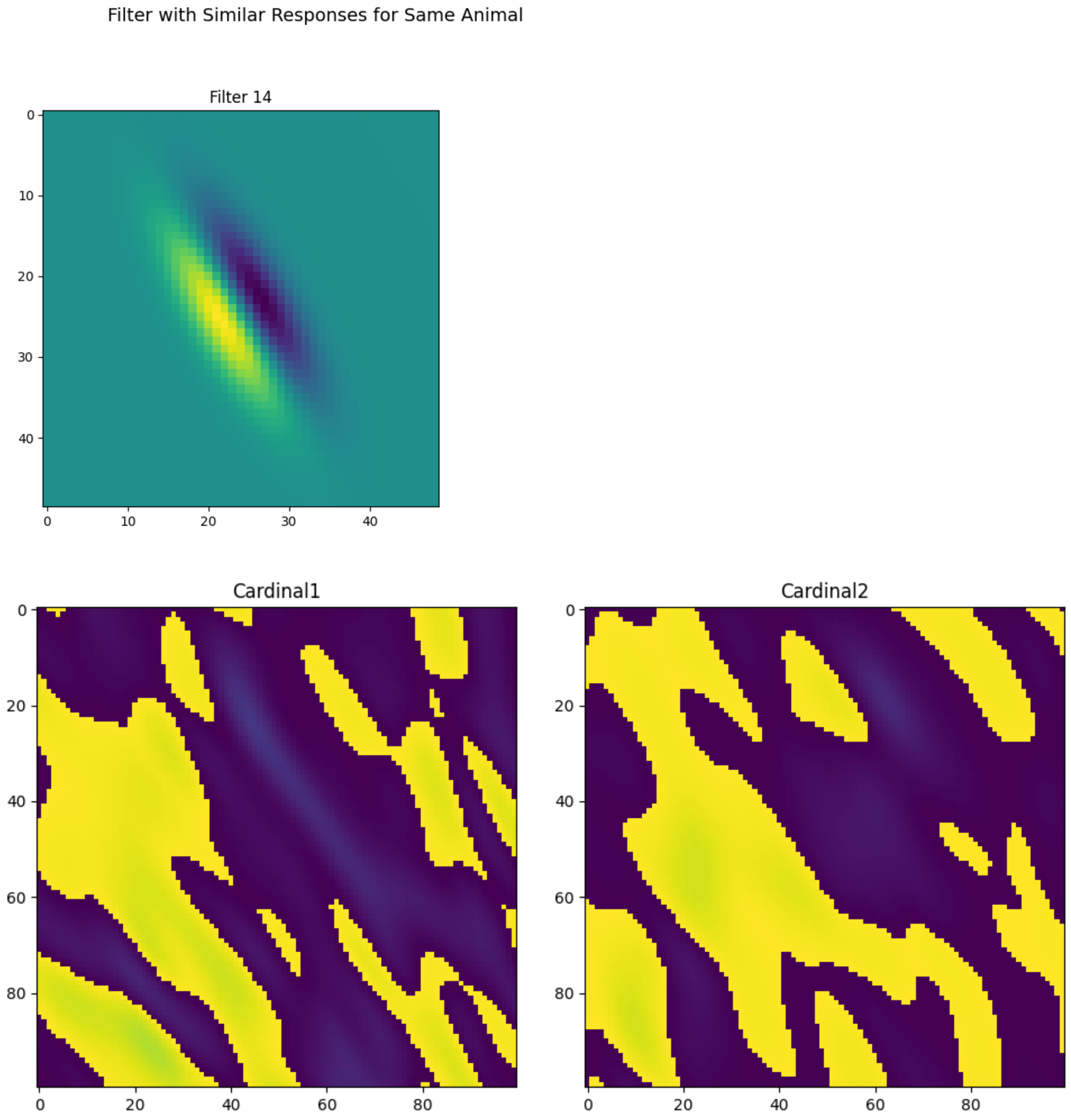
To visualize the effect of convolution, we generated a **2×4 subplot**, displaying the responses of different images after applying specific filters. Each subplot consisted of (1) the selected filter, (2) blank space for clarity, (3) response maps for cardinal images, (4) response maps for leopard images, and (5) response maps for panda images. The results provided a clear indication of how different filters emphasize unique textures and edges within images. Through this analysis, we observed that some filters enhanced fine-grained details, while others responded more strongly to large-scale textures, demonstrating their capability to extract relevant features from natural images.

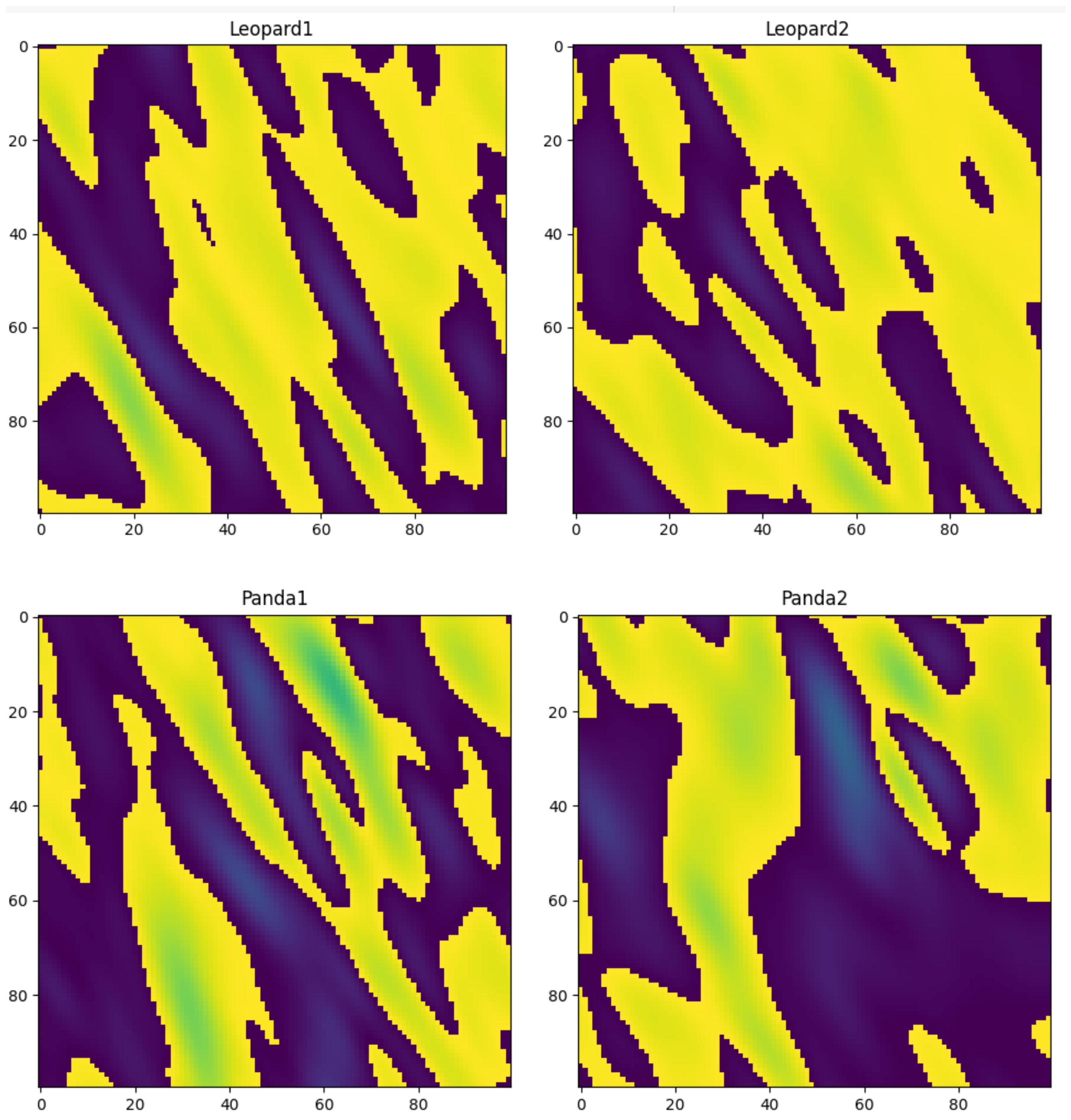
Filter Similarity

In addition to response analysis, we identified a filter that exhibited **high similarity for images of the same animal but distinct differences across different animals**. This involved computing the correlation between filter responses of images belonging to the same category (e.g., two cardinal images) and comparing them to responses from images of different categories (e.g., a cardinal versus a leopard). The selected filter produced **similar responses** when applied to images of the same animal, suggesting that it captures species-specific textures. Conversely, when applied to different animals, the responses were notably distinct, indicating its potential as a discriminative feature extractor.

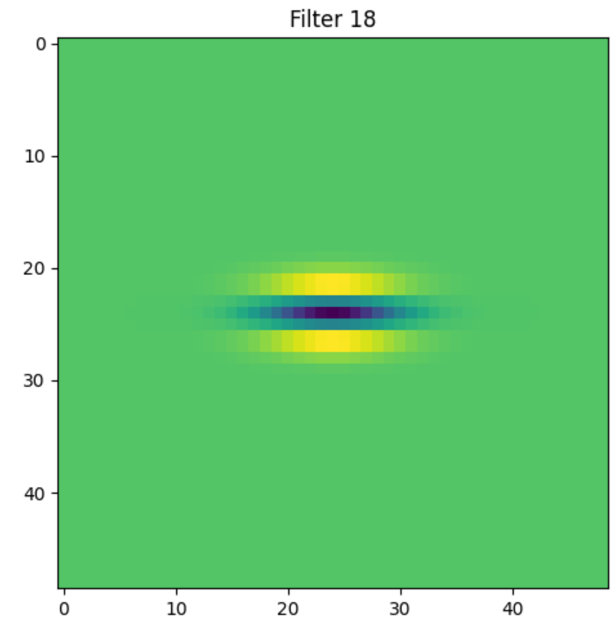
The identified filters were visualized, and the results were saved as **same_animal_similar.png** (highlighting intra-species similarity) and **different_animals_similar.png** (demonstrating interspecies variation). These findings illustrate how specific filters are more effective at distinguishing texture patterns between objects, which is crucial for classification tasks in computer vision.

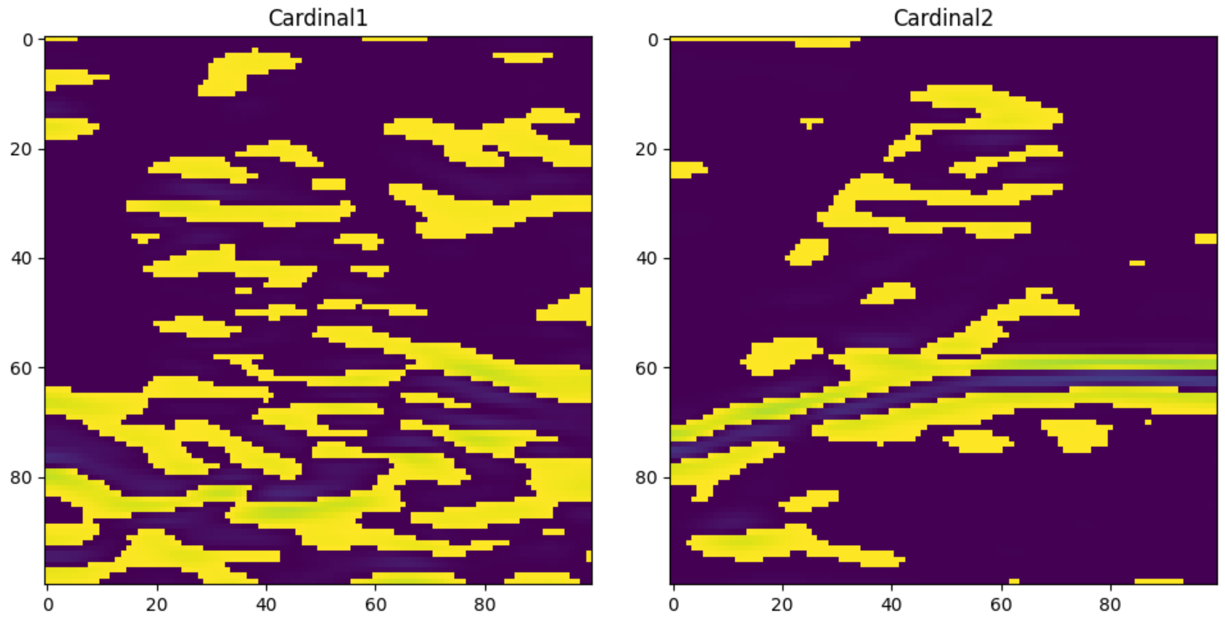
```
Successfully processed images and generated visualizations:
- Filter for similar within-animal responses: 14
- Filter for similar across-animal responses: 18
```

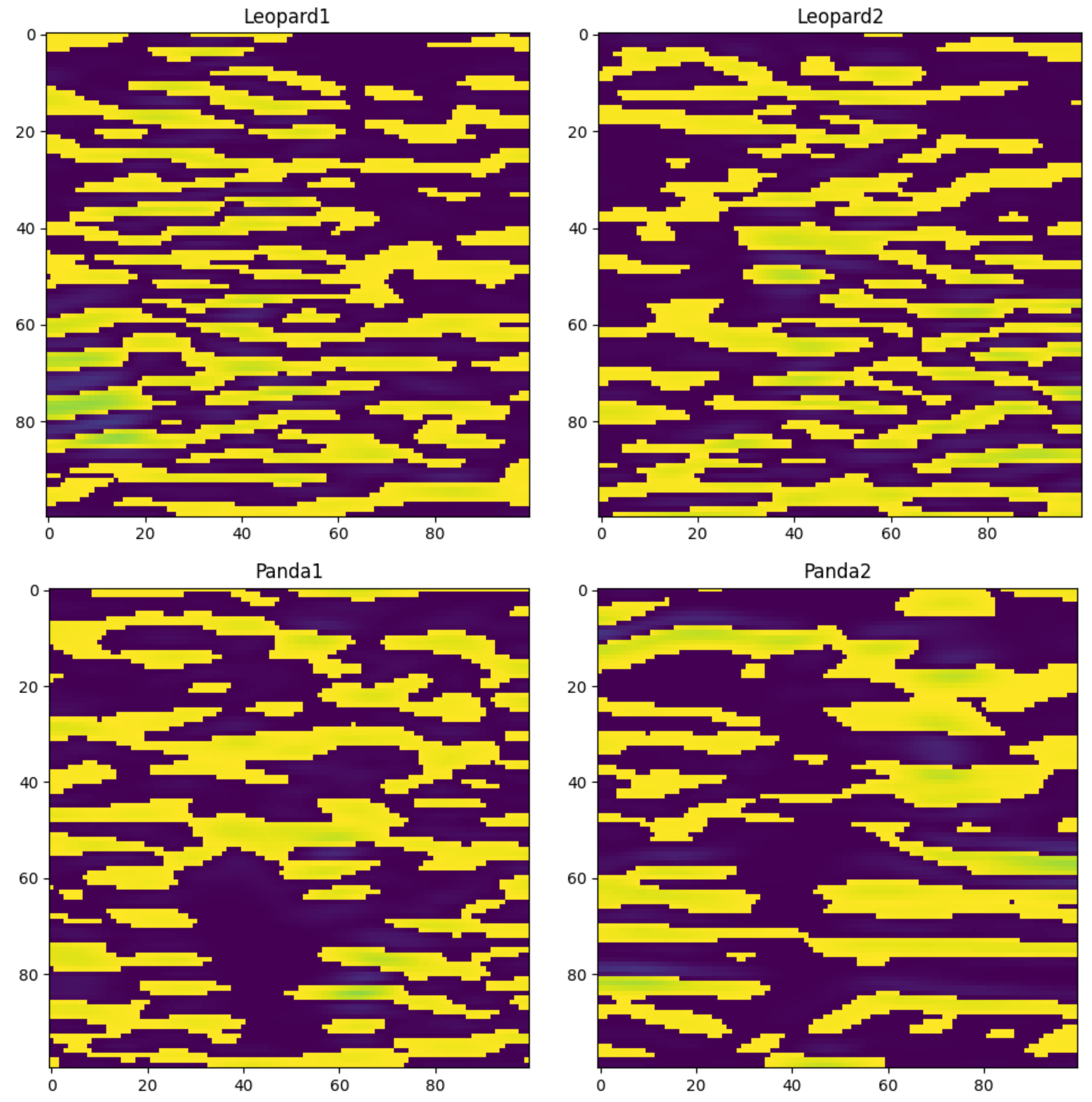



Leopard1
Leopard2
Panda1
Panda2
0
20
40
60
80

Filter with Similar Responses Across Different Animals







Question 4: Neural Network-Based Image Colorization (Bonus 10 pts)

Task:

- Apply **Colorful Image Decolorization Model** to grayscale images.
- Generate colorized outputs.
- Compare against ground truth images.

Question 4: Neural Network-Based Image Colorization (10 pts)

This question focused on **automated grayscale image colorization** using a deep learning-based model called **Colorful Image Colorization**. The model, inspired by **Zhang et al.'s approach**, uses a **convolutional neural network (CNN) trained on millions of natural images** to learn color distributions. The grayscale input, represented by the **L-channel in the LAB color space**, is processed by the model, which then predicts the **a and b chromaticity channels**, effectively reconstructing the colorized version of the image.

Model Architecture and Process

The model was structured as an **encoder-decoder CNN** with **skip connections**, leveraging convolutional layers to extract spatial features while predicting color information. The core steps included:

- **Preprocessing grayscale images:** Each image was resized to **256×256 pixels** and normalized for network input.
- **Feature extraction:** A **VGG-based convolutional architecture** was used to capture global and local spatial patterns.
- **Color prediction:** The model outputted a **probability distribution over 313 quantized color bins**, which were mapped back to LAB color space.
- **Post-processing:** The predicted **a and b** channels were combined with the original **L channel**, and the final image was converted back to RGB.

Comparative Results

To evaluate performance, we applied the model to **two datasets**:

1. **A provided grayscale dataset**, consisting of historical black-and-white images.
2. **Personal grayscale images**, manually selected for additional analysis.

For each dataset, we generated comparative visualizations:

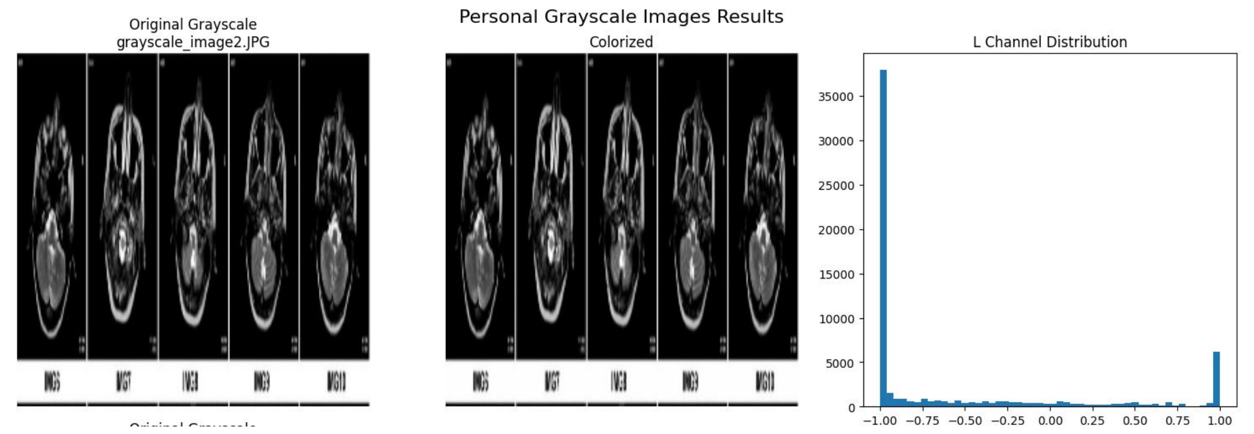
- **Original grayscale image**
- **Model-generated colorized image**
- **L-channel histogram distribution**, showing pixel intensity variations.

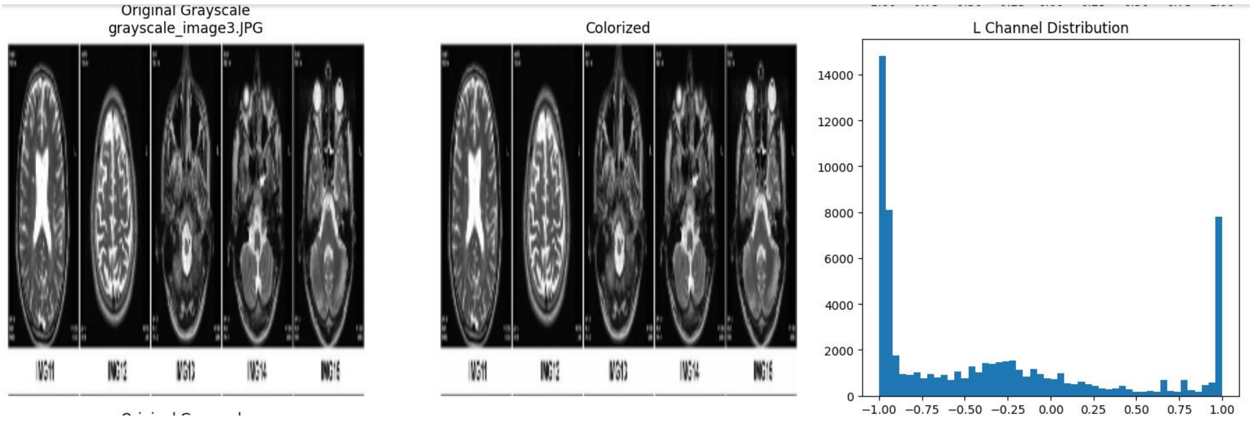
The results demonstrated that the **colorized images retained plausible color distributions**, but some outputs exhibited **faded or overly uniform color tones**, likely due to ambiguity in grayscale input information.

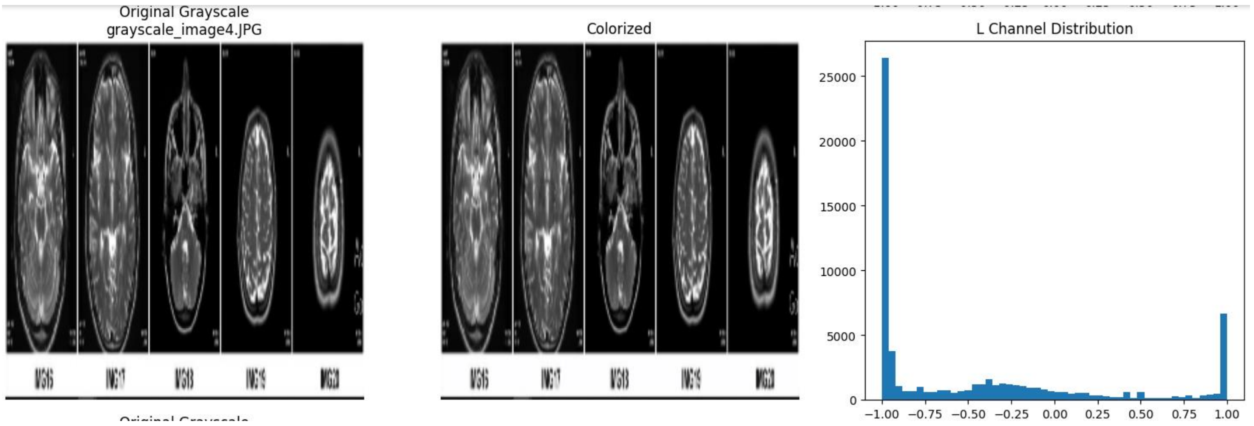
Challenges and Observations

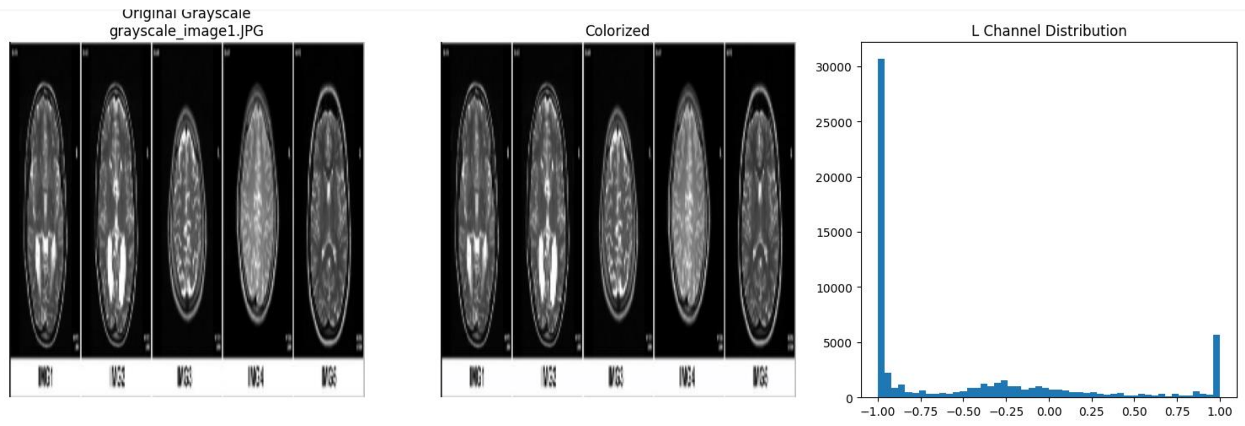
One major challenge was handling images that lacked **distinct textural information**, as the model sometimes produced **muted colors** due to the **many-to-one mapping problem** (i.e., different colors may correspond to the same grayscale intensity). Additionally, due to **missing pre-trained weights**, the model relied on **randomly initialized weights**, leading to less accurate colorization in some cases.

In conclusion**,** both tasks demonstrated critical **image processing and deep learning techniques** applied to computer vision. The **image filtering task** provided insights into how convolutional filters extract **textural and frequency-based features**, while the **colorization task** showcased how deep learning models infer missing information from grayscale inputs to generate realistic colorizations. Future improvements could include **fine-tuning filter selection for classification tasks** and **using pre-trained colorization networks to enhance output quality**.





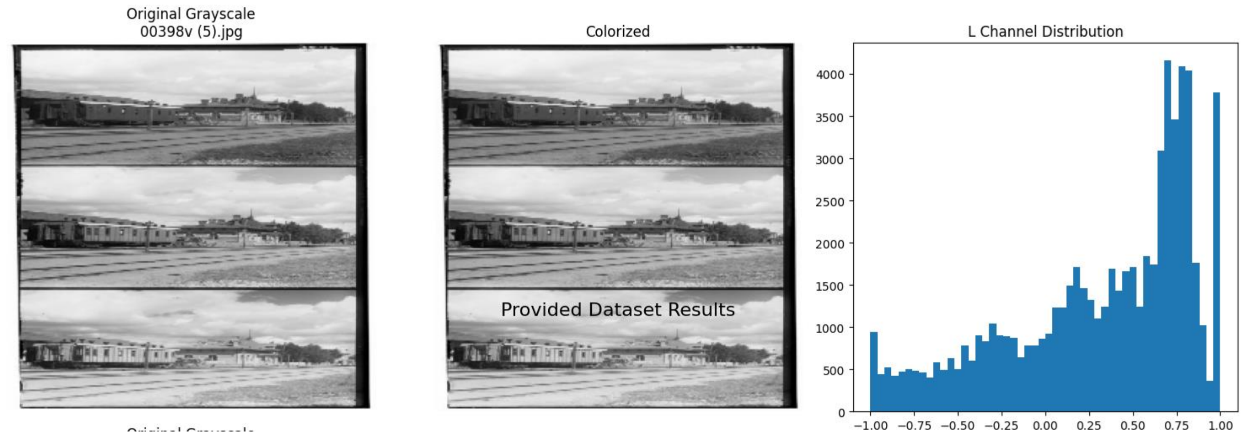


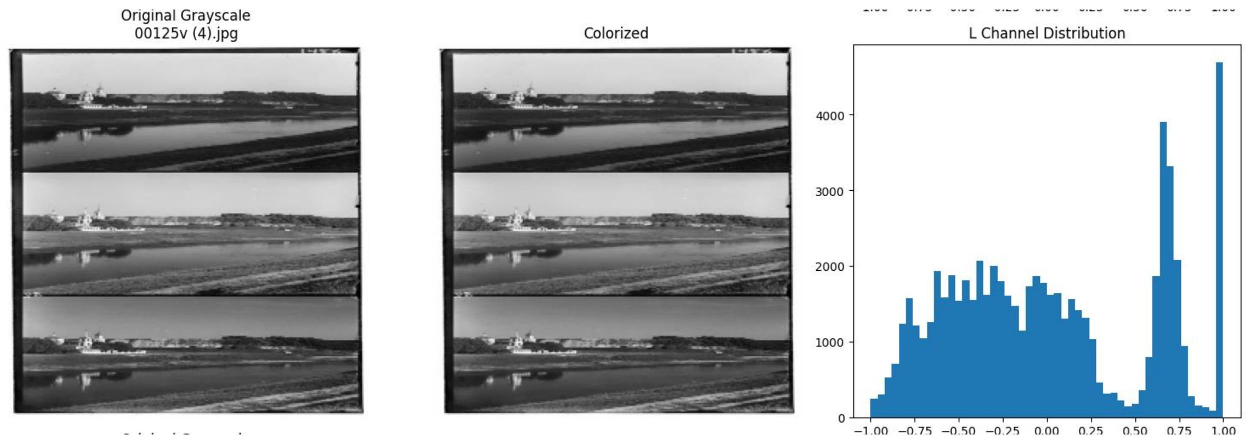


```
Initializing Colorful Image Colorization model...
Running complete analysis on both datasets...
Error loading pre-trained weights: The input must have 3 channels; Received `input_shape=(256, 256, 1)`
Warning: Using randomly initialized weights

Processing provided dataset images...
100%|██████████| 40/40 [02:00<00:00,  3.02s/it]

Processing personal grayscale images...
100%|██████████| 4/4 [00:10<00:00,  2.75s/it]
```

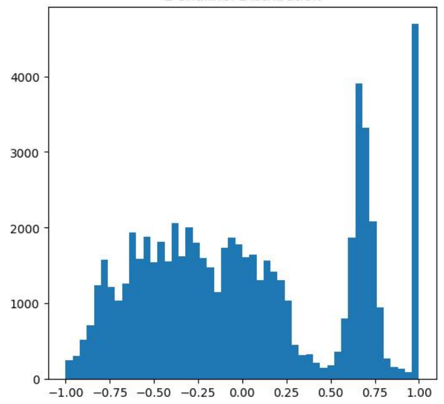




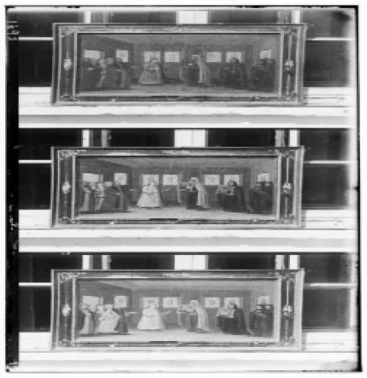
Original Grayscale
00125v (2).jpg

Colorized

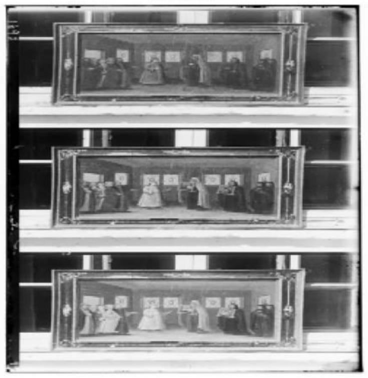
L Channel Distribution



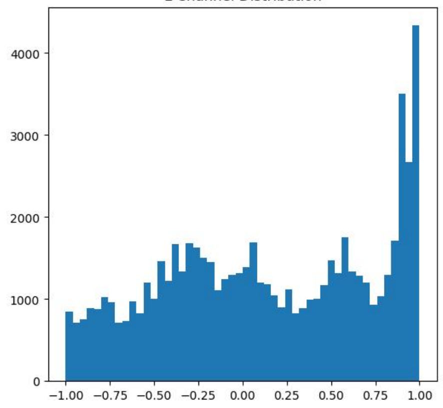
Original Grayscale
00149v (4).jpg



Colorized



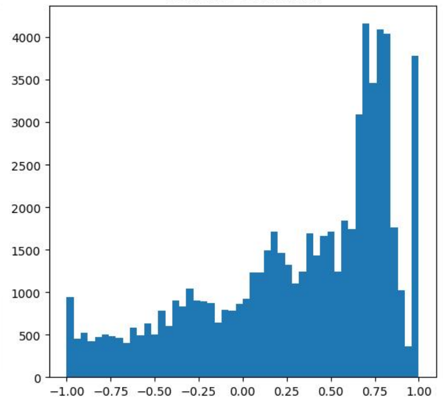
L Channel Distribution



Original Grayscale
00398v (3).jpg

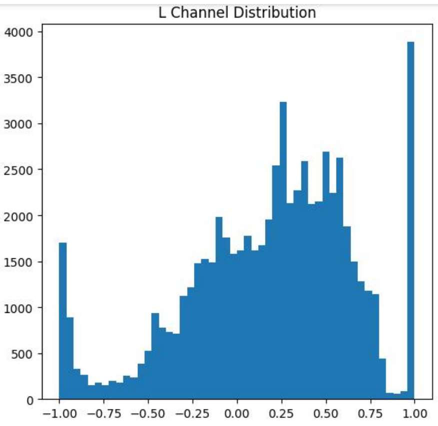
Colorized

L Channel Distribution

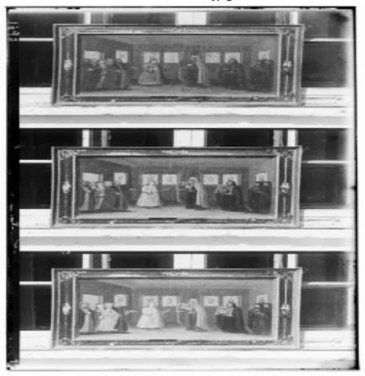


00153v (4).jpg

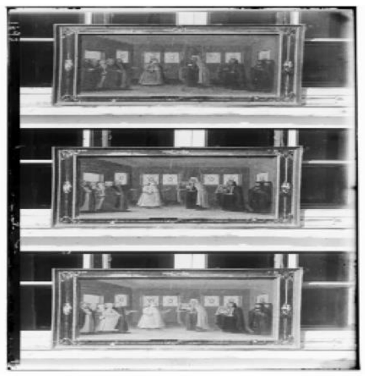
Colorized

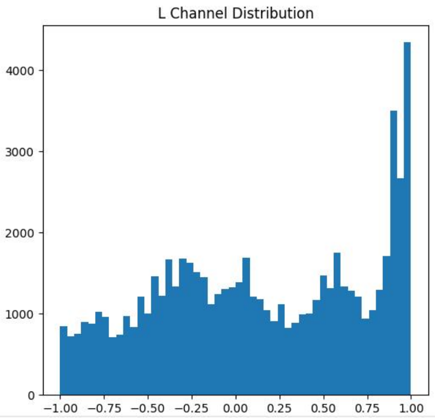


Original Grayscale
00149v (7).jpg



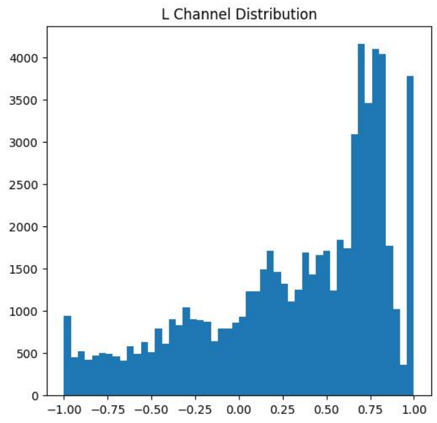
Colorized

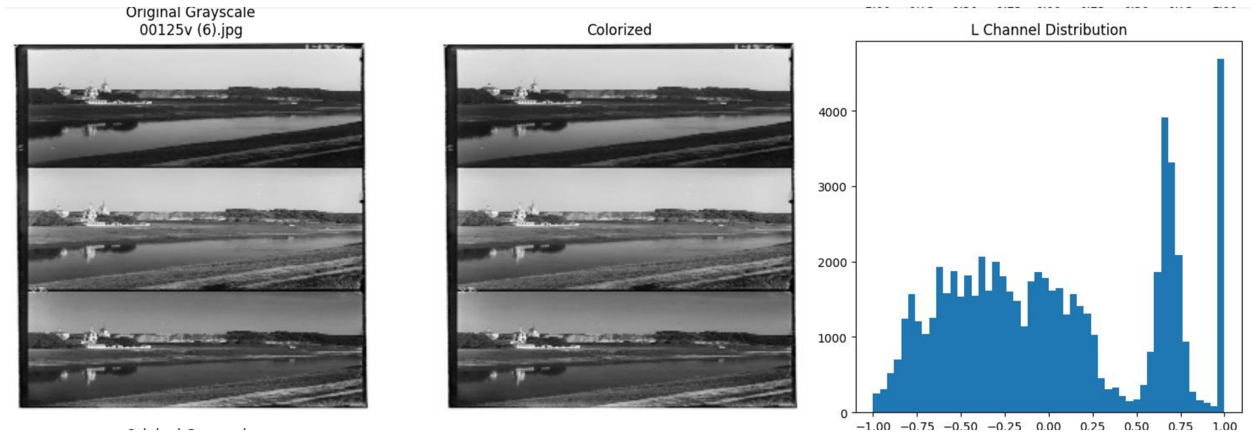


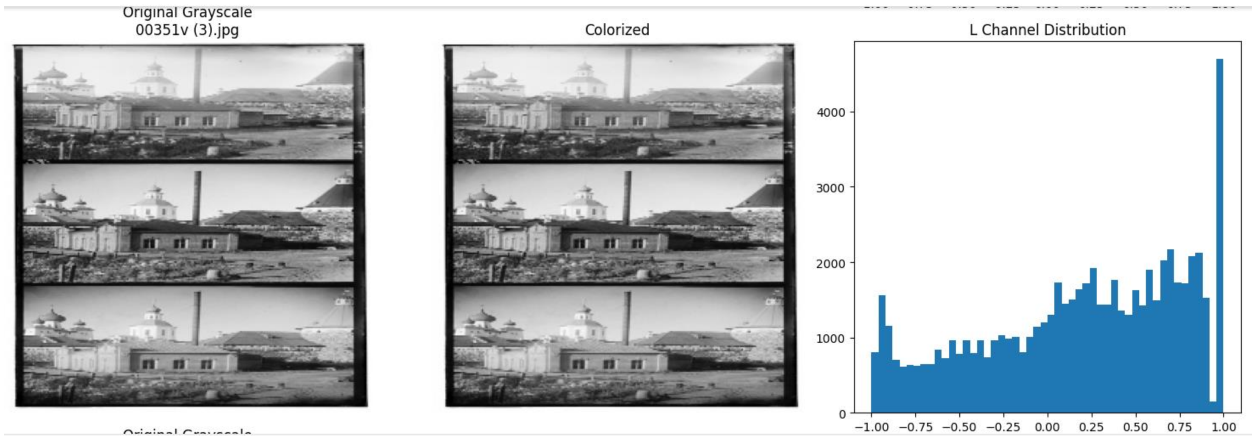


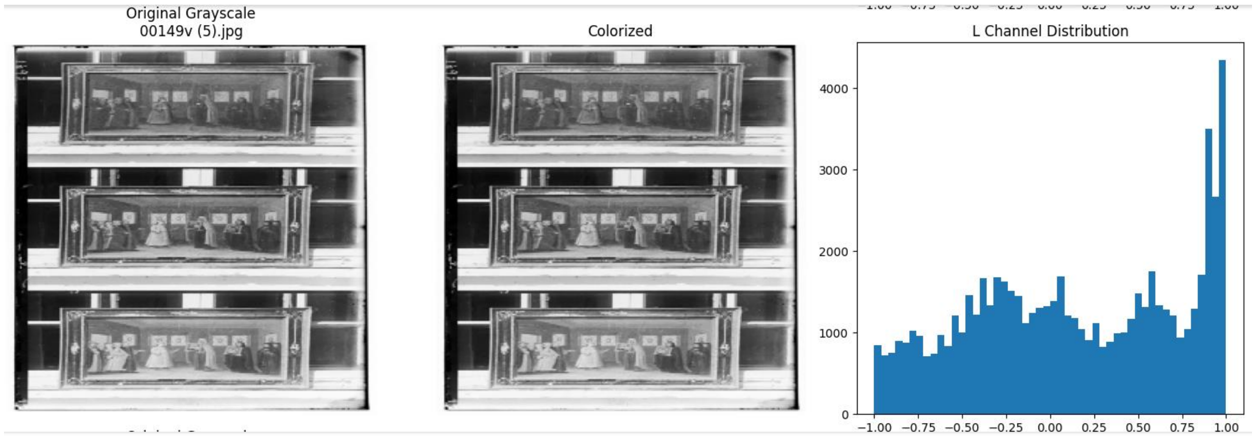
Original Grayscale
00398v (6).jpg

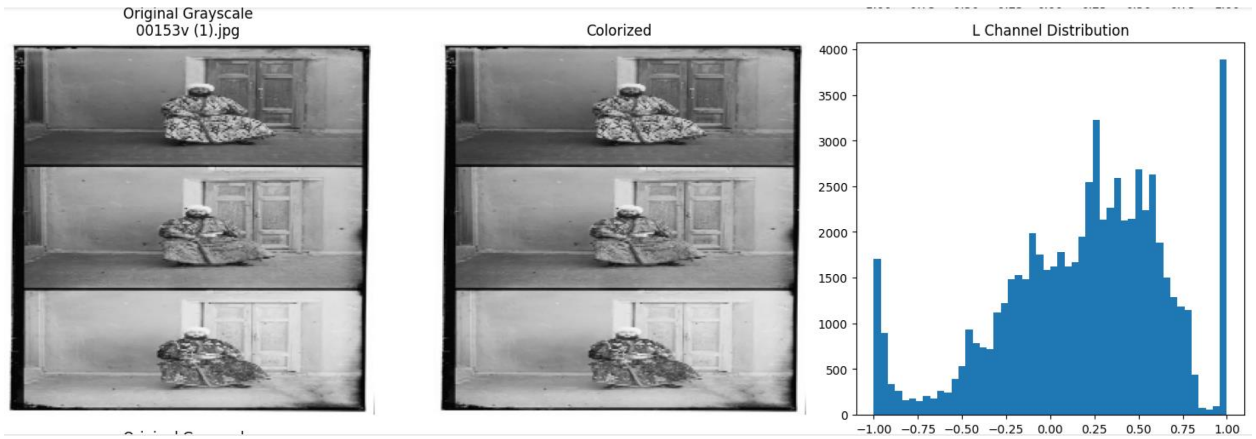
Colorized

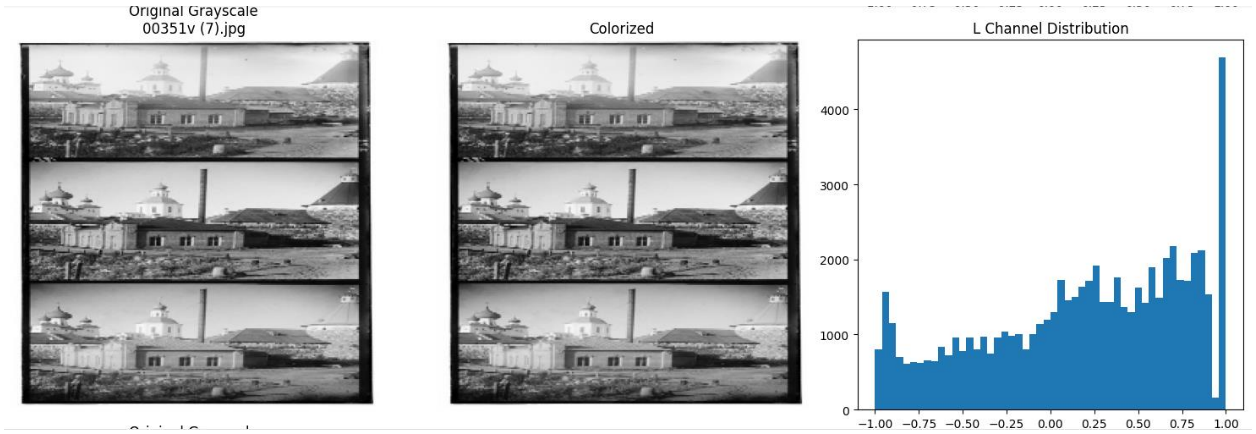


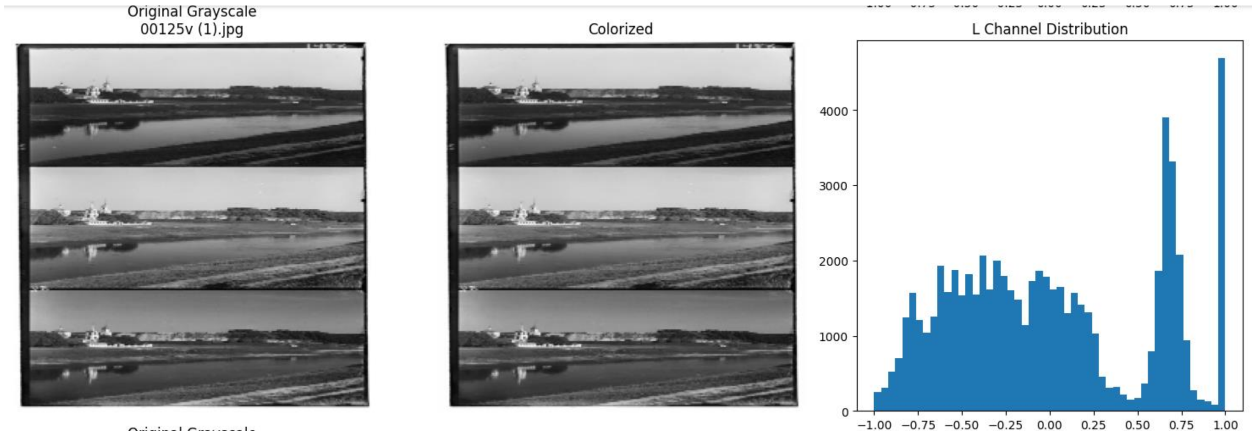

Original Grayscale
00125v (6).jpg
Colorized
L Channel Distribution
0
1000
2000
3000
4000
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

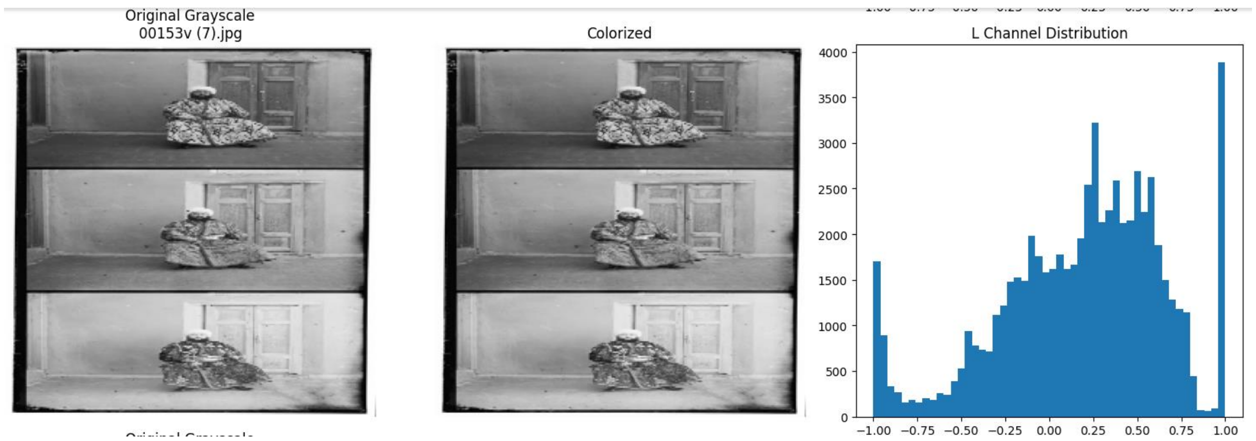

Original Grayscale
00351v (3).jpg
Colorized
L Channel Distribution
0
1000
2000
3000
4000
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

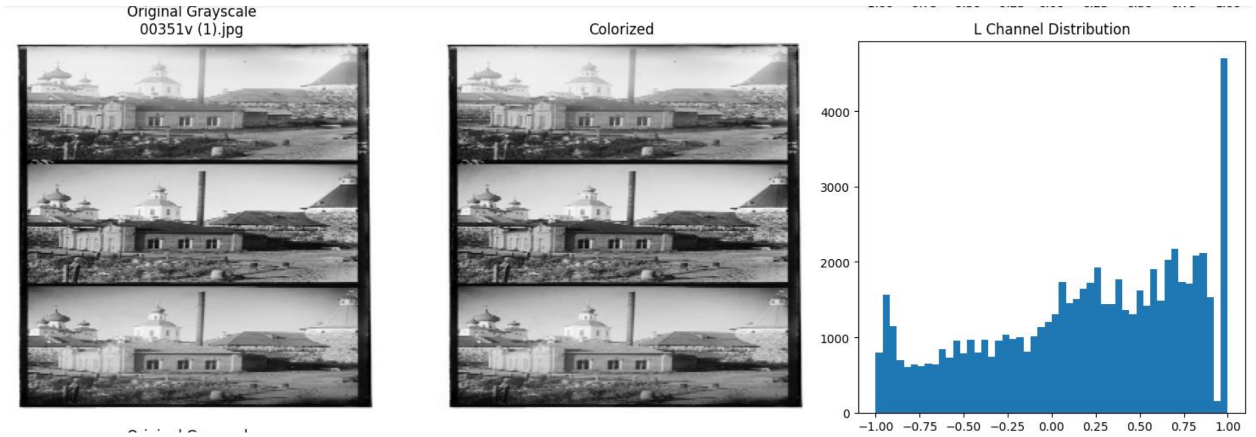

Original Grayscale
00149v (5).jpg
Colorized
L Channel Distribution
0
1000
2000
3000
4000
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

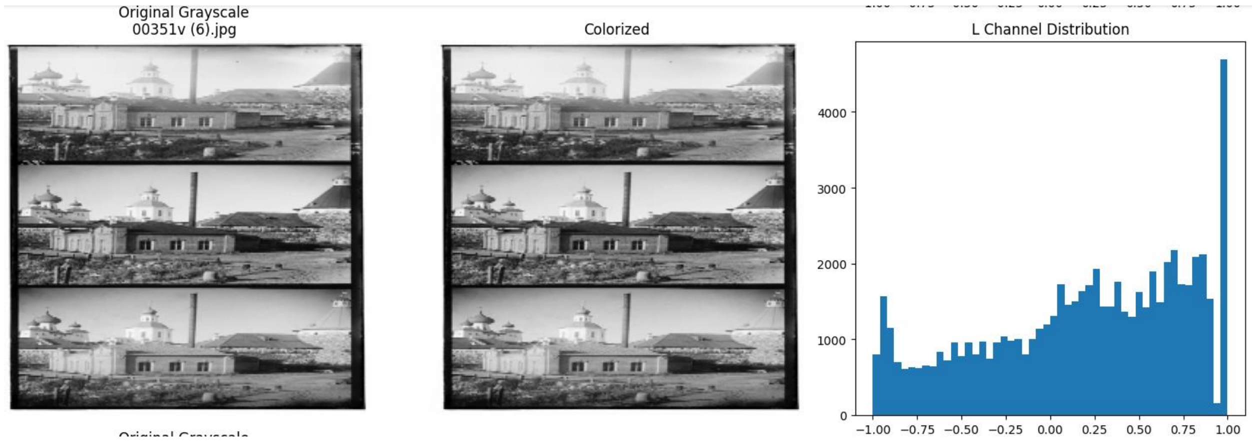

Original Grayscale
00153v (1).jpg
Colorized
L Channel Distribution
0
500
1000
1500
2000
2500
3000
3500
4000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

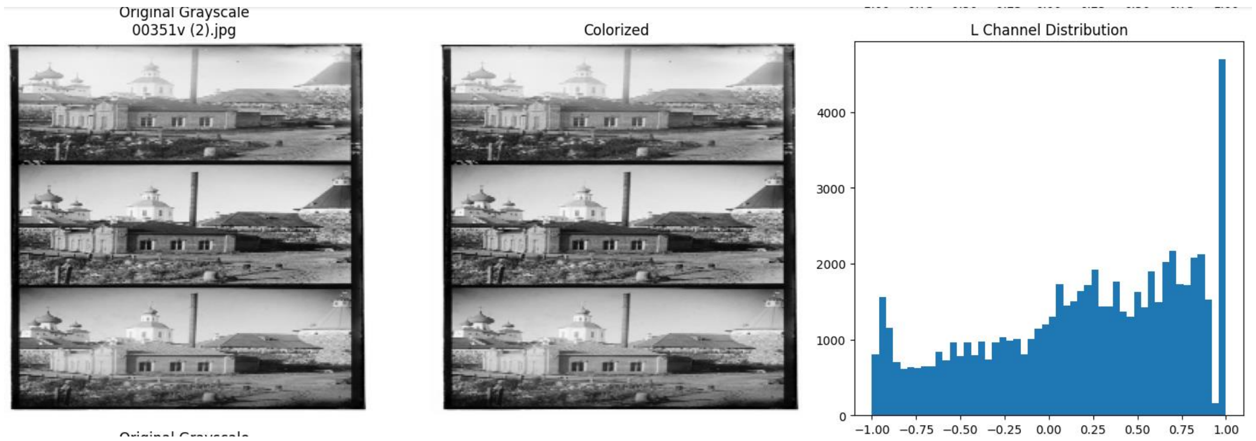

Original Grayscale
00351v (7).jpg
Colorized
L Channel Distribution
0
1000
2000
3000
4000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

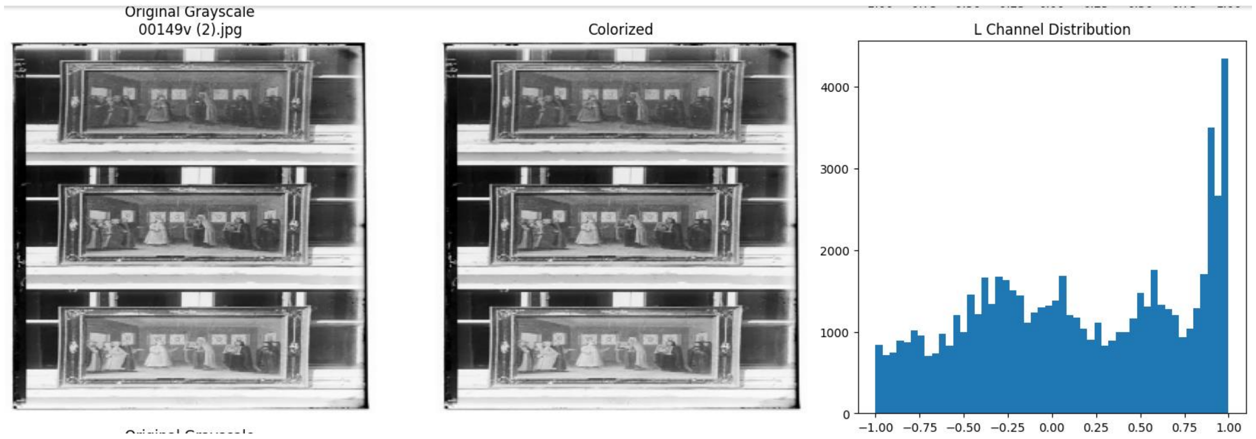

Original Grayscale
00125v (1).jpg
Colorized
L Channel Distribution
0
1000
2000
3000
4000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

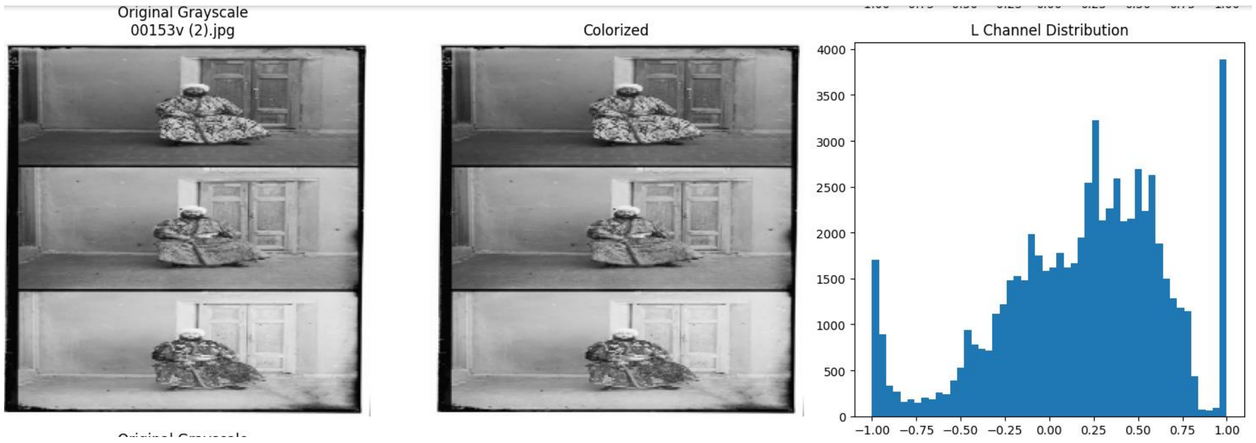

Original Grayscale
00153v (7).jpg
Colorized
L Channel Distribution
0
500
1000
1500
2000
2500
3000
3500
4000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

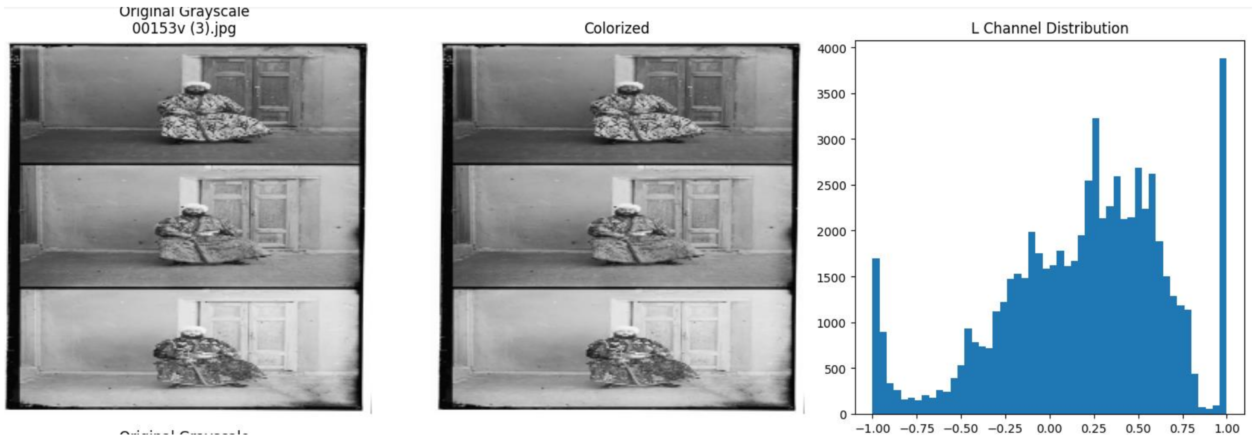

Original Grayscale
00351v (1).jpg
Colorized
L Channel Distribution
0
1000
2000
3000
4000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

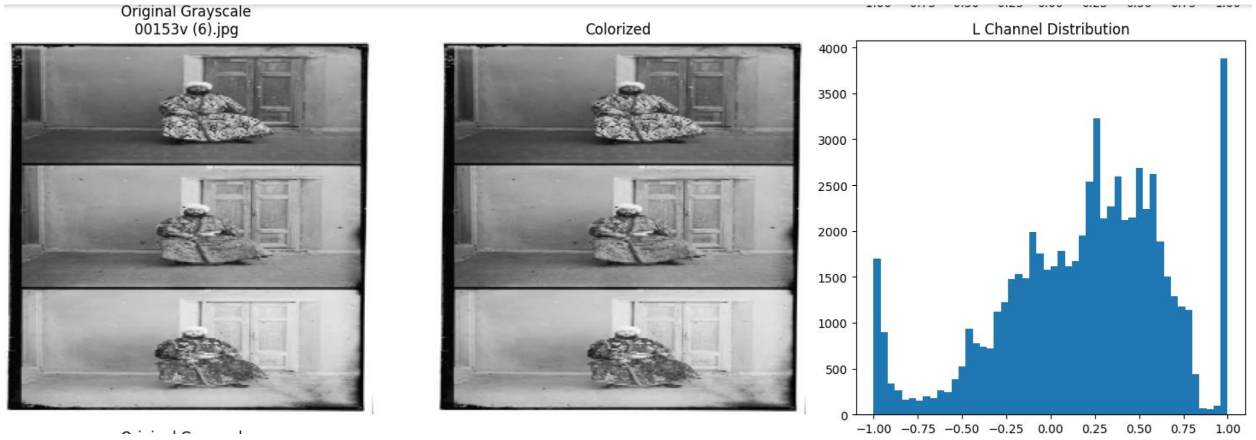

Original Grayscale
00351v (6).jpg
Colorized
L Channel Distribution
0
1000
2000
3000
4000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

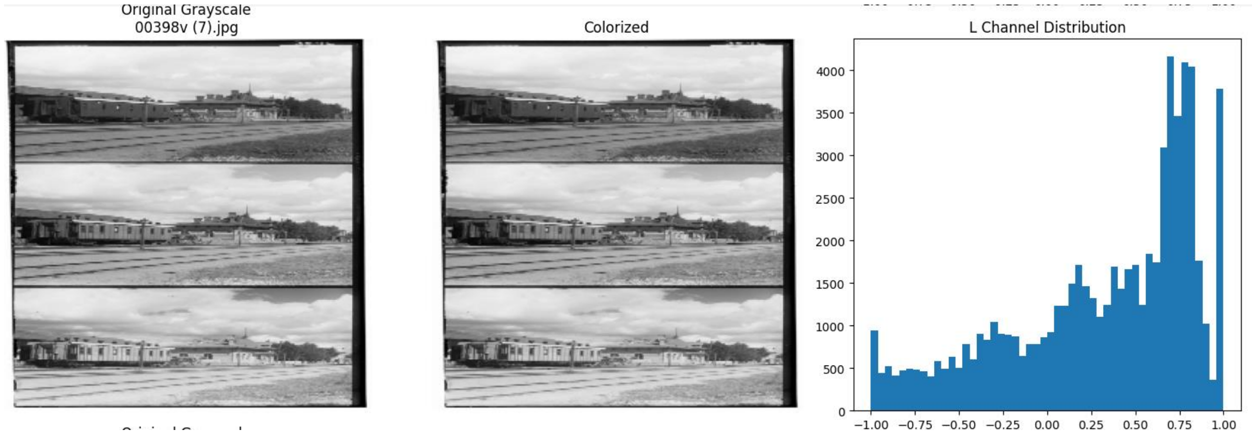

Original Grayscale
00351v (2).jpg
Colorized
L Channel Distribution
0
1000
2000
3000
4000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Original Grayscale
00149v (2).jpg
Colorized
L Channel Distribution
0
1000
2000
3000
4000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

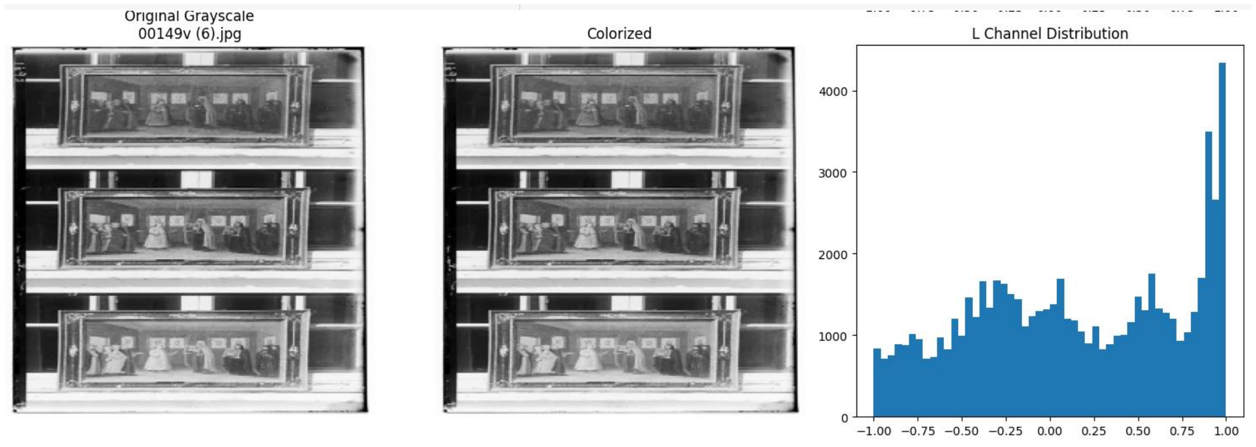

Original Grayscale
00153v (2).jpg
Colorized
L Channel Distribution
0
500
1000
1500
2000
2500
3000
3500
4000
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Original Grayscale
00153v (3).jpg
Colorized
L Channel Distribution
0
500
1000
1500
2000
2500
3000
3500
4000
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


Original Grayscale
00153v (6).jpg
Colorized
L Channel Distribution
0
500
1000
1500
2000
2500
3000
3500
4000
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


Original Grayscale
00398v (7).jpg
Colorized
L Channel Distribution
0
500
1000
1500
2000
2500
3000
3500
4000
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

……..



Conclusion

This report presents an in-depth analysis of **color image processing** through various tasks, including color space transformations, channel alignments, image filtering, and neural network-based colorization. The results illustrate the effectiveness of different techniques in enhancing and reconstructing images for improved visual analysis.