COL783 A1

Anuj Naval (2021CS50591) & Harshit Kumar Gautam (2021CS50129)

August 2025

1 Part 1 - Document Scanner

1.1 Document Rectification and Interpolation

The goal of this task is to implement a perspective transformation to straighten a photographed document and make it appear as if it were scanned. This is similar to the functionality provided by document scanner applications.

First, we take an image of a document with all four corners visible and slightly tilted to ensure that the document is not perfectly vertical. The first step involves detecting or selecting the four corner points of the document. We manually select these points using a mouse interaction method. The selected points are shown in the image below:

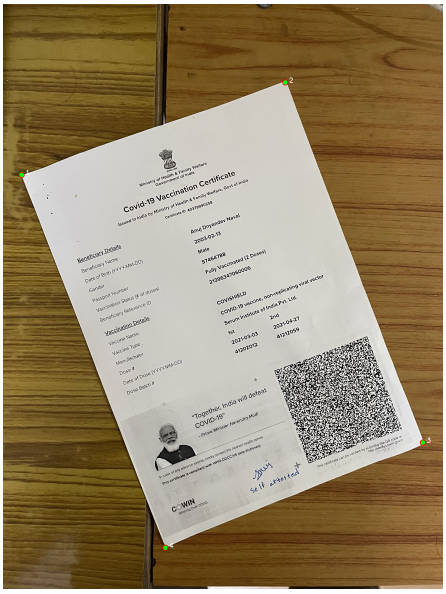


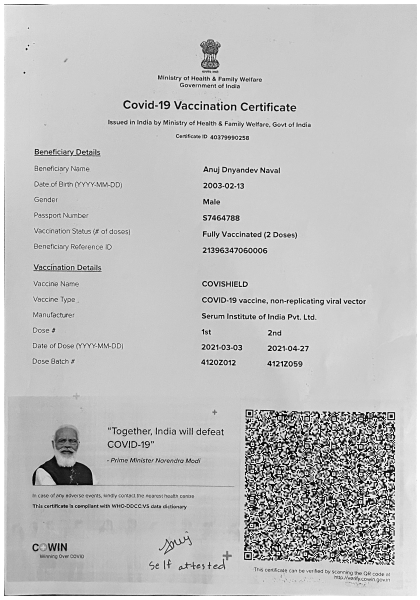
Figure 1: Original image with selected four corner points.

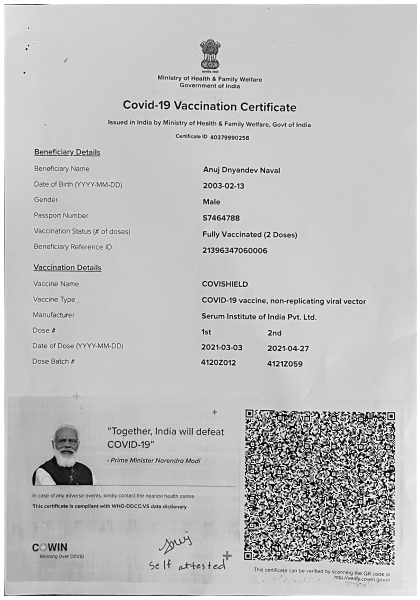
Once we have the four corner points, we compute a *homography matrix* that maps these points to a rectangle representing the top-down view of the document. The destination rectangle is chosen so that it preserves the document's aspect ratio and contains as little background as possible.

The homography is applied using two interpolation methods:

- **Nearest Neighbor Interpolation:** This method assigns the value of the nearest pixel from the source image. It is computationally cheap but may result in jagged edges.
- **Bilinear Interpolation:** This method computes the pixel value by taking a weighted average of the four nearest neighbors, resulting in a smoother appearance.

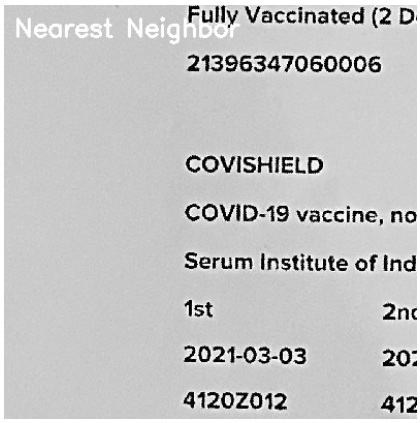
The outputs of both interpolation methods are shown below:



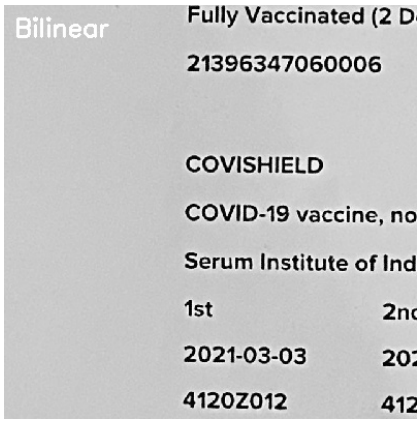


Left: Nearest Neighbor Interpolation *Right: Bilinear Interpolation*

To clearly observe the difference between the two methods, we also zoom in on the center region of the document:



Left: Nearest Neighbor (Zoomed)



Right: Bilinear (Zoomed)

From the zoomed-in images, it is evident that bilinear interpolation provides smoother and less pixelated results compared to nearest neighbor interpolation.

Procedure Summary:

1. Manually select four corner points of the document in the input image.
2. Compute the homography matrix to map the selected quadrilateral to a rectangle.
3. Apply the transformation using both nearest neighbor and bilinear interpolation.
4. Compare the results and analyze differences using zoomed-in views.

1.2 Contrast Enhancement Techniques

Typically, a photograph of a printed page does not have high enough contrast, making the background appear gray and the text appear faded. To address this issue, we implemented several techniques to enhance the contrast so that the paper becomes nearly white and the text becomes nearly black. The following methods were applied:

Manual Thresholding

The simplest approach to enhance contrast is by applying a manually chosen intensity threshold. In this method, all pixels with intensity values greater than the threshold are set to white (255), and all others to black (0), resulting in a binary image. For our implementation, we used a threshold value of 127. This method is computationally inexpensive but is highly sensitive to the chosen threshold and may lose text details in low-contrast regions.

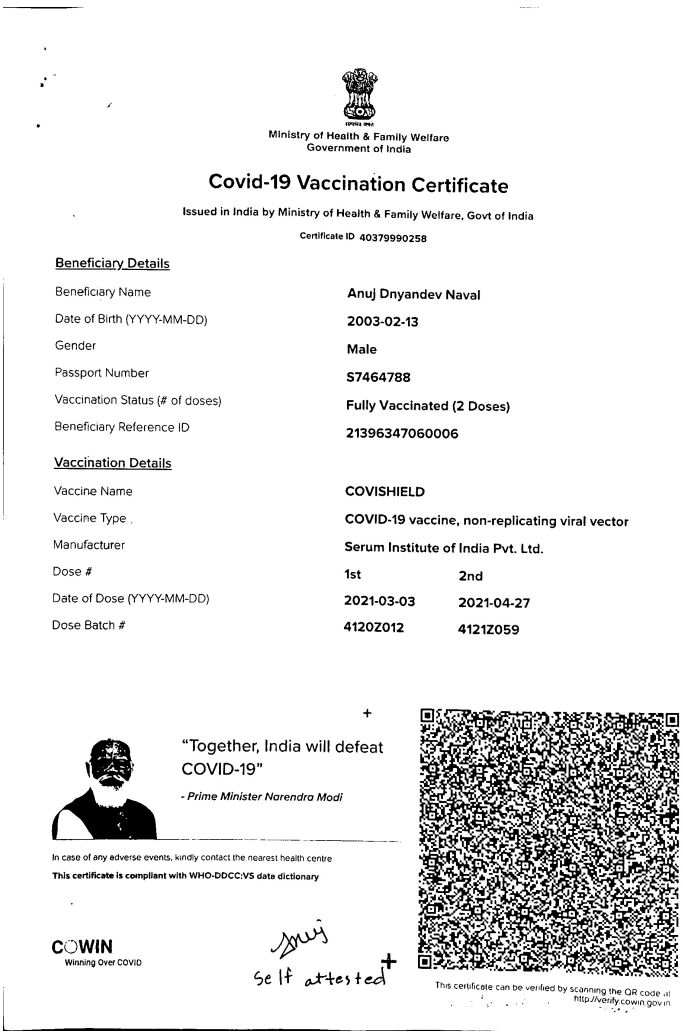


Figure 2: Result of manual thresholding.

Manual Linear Intensity Transformation

A more flexible approach is to apply a linear transformation of the form:

$$T(r) = a \cdot r + b$$

where a controls the contrast and b controls the brightness. We manually selected $a = 1.5$ and $b = -50$ to achieve the best possible contrast for our image. This method works better than thresholding because it preserves grayscale variations while enhancing the text contrast.

Figure 3: Result of manual linear intensity transformation.

Automatic Linear Intensity Transformation

Instead of selecting parameters manually, we computed them automatically based on the image histogram. Specifically, we used the 5th and 95th percentiles of the grayscale histogram to determine the lower and upper bounds of intensity values. This approach effectively stretches the dynamic range of the image such that:

$$a = \frac{255}{P_{95} - P_5}, \quad b = -a \cdot P_5$$

where P_5 and P_{95} represent the 5th and 95th percentile intensities. This ensures that most of the pixel values are mapped to the full range [0, 255], improving

contrast in a data-driven manner.

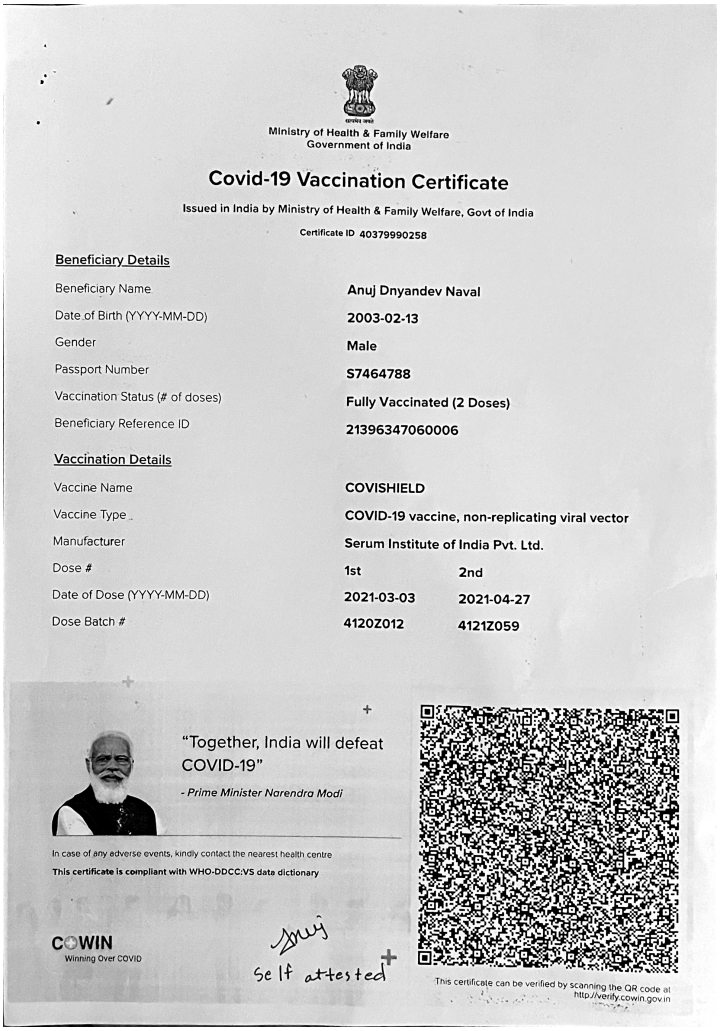


Figure 4: Result of automatic linear intensity transformation.

Histogram Equalization

Finally, we applied histogram equalization, which redistributes the pixel intensities so that the histogram becomes approximately uniform. We applied this method on the Value channel in the HSV color space to preserve color balance. Histogram equalization significantly improves global contrast, but in our case, the result appears visually noisy because the document background contains uneven illumination. The algorithm enhances not only the text but also the background variations, leading to artifacts.

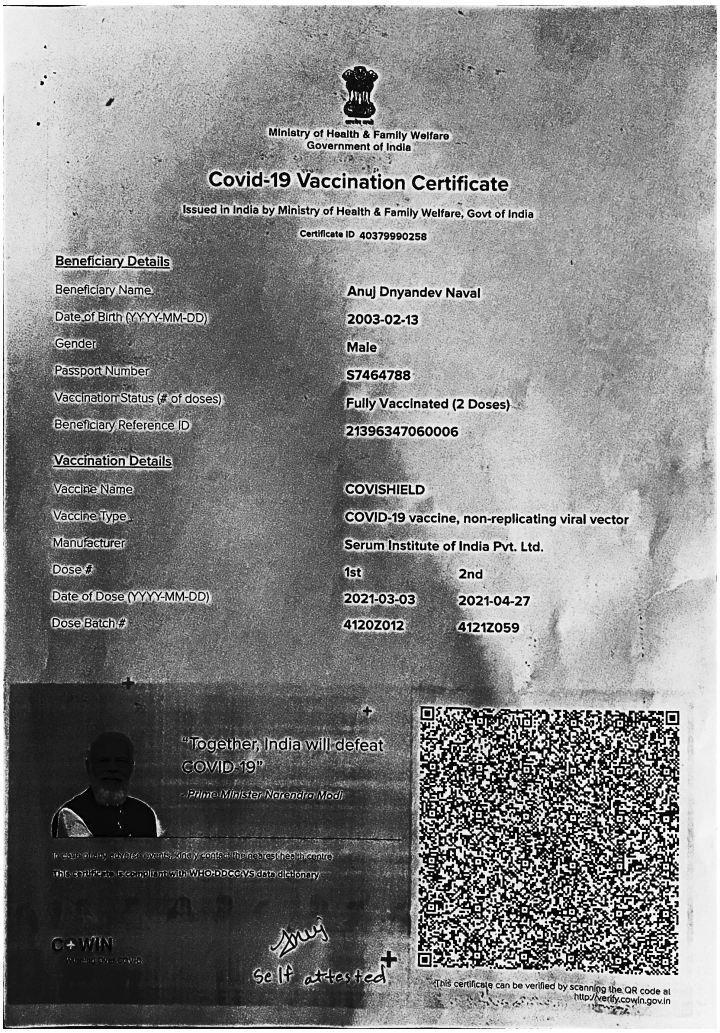


Figure 5: Result of histogram equalization. Notice that noise and background patterns are amplified.

Observation: - Manual thresholding works well when the text and background are clearly separable but fails for uneven lighting. - Manual linear transformation gives better results but requires manual tuning. - Automatic linear transformation performs well in general by adapting to the image histogram. - Histogram equalization enhances global contrast but also amplifies noise and background artifacts, making it less suitable for clean document scanning.

1.3 Watermarking the Document

After enhancing the contrast of the scanned document, the next step is to add a watermark to simulate a premium feature of a document scanning application. The watermark used is the IIT Delhi logo.

Logo Preprocessing

The IIT Delhi logo initially contains a white background, which needs to be removed before overlaying it on the document. The steps are as follows:

1. Convert the logo image to grayscale using:

$$\text{Gray}(x, y) = 0.299R + 0.587G + 0.114B$$

2. Apply thresholding to create a binary mask that separates the logo from the white background:

$$m(x, y) = \begin{cases} 255, & \text{if Gray}(x, y) \leq T \\ 0, & \text{otherwise} \end{cases}$$

   where $T = 200$ in our implementation.

The original logo, mask, and resized logo are shown in Figure 6.

Left: Original Logo *Center: Logo Mask* *Right: Resized Logo*

Figure 6: Logo preprocessing steps.

Logo Resizing

The watermark should have a width equal to 20% of the document width. Let W_d be the document width and W_l, H_l the original logo dimensions. The new size is computed as:

$$W_l' = 0.2 \times W_d, \quad H_l' = \frac{H_l}{W_l} \times W_l'$$

The logo and mask are resized using bilinear interpolation.

Applying the Watermark

The resized logo is placed at the bottom-right corner of the document with 50% transparency. Let:

- $d(x, y)$: pixel intensity of the document
- $l(x, y)$: pixel intensity of the logo
- $m(x, y)$: binary mask (1 for logo, 0 for background)

The blending formula is:

$$\text{output}(x, y) = (1 - \alpha m(x, y))d(x, y) + (\alpha m(x, y))l(x, y)$$

with $\alpha = 0.5$. For mask values, it simplifies to:

$$\text{output}(x, y) = \begin{cases} 0.5d(x, y) + 0.5l(x, y), & \text{if } m(x, y) = 1 \\ d(x, y), & \text{if } m(x, y) = 0 \end{cases}$$

The final watermarked document is shown in Figure 7.

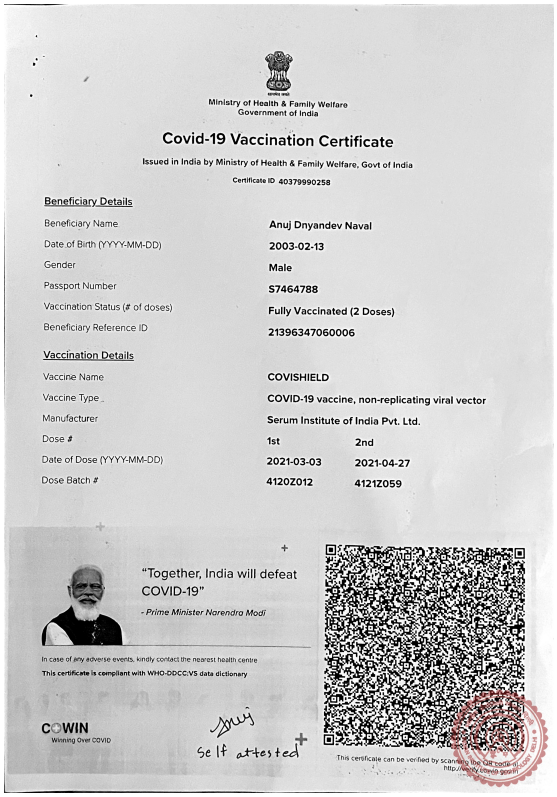


Figure 7: Document image with IIT Delhi logo watermark applied at the bottom-right corner.

Observations

- The white background of the logo is removed using the mask, preserving the document text.
- Transparency ensures that the watermark does not hinder readability.
- Resizing based on document width makes the watermark proportional for different document sizes.

1.4 High Dynamic Range (HDR) Image Processing

Histogram Equalization for HDR Images Without Quantization

High Dynamic Range (HDR) images capture a much wider range of luminance values than standard 8-bit images. In such images, pixel intensities can be arbitrary real numbers rather than discrete quantized values. This poses a challenge for traditional histogram equalization, which relies on building a frequency table of size equal to the number of discrete intensity levels. For HDR images, this approach becomes impractical due to the lack of quantization.

Objective

The goal is to perform histogram equalization on an HDR image while:

- Handling arbitrary real-valued pixel intensities (no quantization).
- Mapping luminance values to a target range $[0, 255]$ for display.
- Preserving the color balance of the original image.

Algorithm Description

The proposed algorithm avoids quantization by operating directly on the sorted intensity values and computing the cumulative distribution function (CDF) empirically.

1. **Convert to Luminance:** Compute luminance using the standard formula:
$$L(x, y) = 0.299R + 0.587G + 0.114B$$
where R, G, B are the red, green, and blue channel intensities.
2. **Sort Intensities:** Flatten the luminance matrix into a 1D array and sort the values in ascending order while keeping track of the original pixel positions.
3. **Compute Empirical CDF:** For each pixel, compute its rank r among all pixels. The cumulative probability is:
$$p = \frac{r}{N - 1}$$

where N is the total number of pixels.

4. **Map to Target Range:** The equalized intensity is:

$$L'(x, y) = \text{target}_{\min} + p \times (\text{target}_{\max} - \text{target}_{\min})$$

In this case, $\text{target}_{\min} = 0$ and $\text{target}_{\max} = 255$.

5. **Preserve Color Ratios:** For each pixel, compute the scaling factor:

$$s = \frac{L'(x, y)}{L(x, y)}$$

and apply it to all color channels:

$$(R', G', B') = s \cdot (R, G, B)$$

Results

The algorithm was applied to an HDR image from the Paul Debevec dataset (`memorial.hdr`). The original HDR image was tone-mapped for visualization, and the histogram-equalized version was generated by mapping intensities to the range $[0, 255]$.

Left: Original HDR image (tone-mapped) *Right: After histogram equalization*

Figure 8: HDR image before and after histogram equalization without quantization.

Observations

- The equalized image has improved contrast across dark and bright regions.
- The algorithm does not rely on discretizing intensity values, making it suitable for floating-point HDR data.
- Color fidelity is maintained by scaling RGB values based on the luminance ratio.
- The method has a computational cost of $O(N \log N)$ due to sorting, but avoids memory overhead associated with large histograms.

Proof: Invariance of Histogram Equalization under Strictly Increasing Transformations

Statement: If a strictly increasing intensity transformation $T(r)$ is applied to an image before histogram equalization, the result is identical to applying histogram equalization directly to the original image. Moreover, performing histogram equalization twice has no additional effect.

Proof: Let the original intensity range be $r \in [0, L-1]$ and its normalized histogram be $p_r(r)$. The cumulative distribution function (CDF) is:

$$s = G(r) = \int_0^r p_r(w)\, dw$$

where $G(r)$ maps the original intensity r to the equalized intensity $s \in [0, 1]$.

Now, suppose a strictly increasing transformation $T(r)$ is applied before equalization. The new intensity is:

$$z = T(r), \quad \text{with inverse } r = T^{-1}(z)$$

The PDF of z is given by:

$$p_z(z) = p_r(T^{-1}(z)) \cdot \frac{d}{dz} T^{-1}(z)$$

The CDF after transformation is:

$$F(z) = \int_0^z p_z(t)\, dt$$

Substituting $t = T(w)$, $dt = T'(w)\, dw$, and $T^{-1}(t) = w$, we have:

$$F(z) = \int_0^z p_r(T^{-1}(t)) \cdot \frac{d}{dt} T^{-1}(t)\, dt = \int_{w=0}^{w=T^{-1}(z)} p_r(w)\, dw$$

But the right-hand side is simply $G(T^{-1}(z))$. Thus:

$$F(z) = G(T^{-1}(z))$$

Therefore, the equalized intensity after applying $T(r)$ and then histogram equalization is:

$$s' = F(z) = G(T^{-1}(z)) = G(r)$$

which is identical to the original equalization result.
Consequences:

1. Any strictly increasing transformation does not change the final result of histogram equalization.

2. Applying histogram equalization twice has no additional effect, since the mapping after the first equalization is already uniform ($s = G(r)$), making the second equalization an identity transformation.

□

Part 2: Convolution and Gaussian Filtering

2 Task 5(a): Laplacian Filtering with Bias

We implemented a custom convolution function that accepts floating-point kernels and 8-bit images. A 3×3 Laplacian kernel was applied with bias $c = 128$. The bias shifts the output to the visible $[0, 255]$ range, avoiding clipping of negative values.

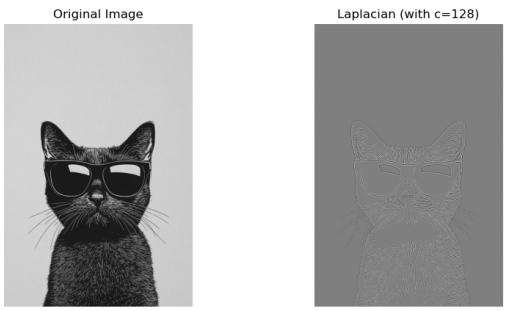


Figure 9: Original image and Laplacian filtered result ($c = 128$).

3 Task 5(b): Gaussian Filtering (Naive vs Separable)

Gaussian smoothing was performed in two ways:

1. Direct 2D convolution with Gaussian kernel.

2. Exploiting separability into two 1D convolutions (row then column).

We used $\sigma = 5.0$ with kernel size $k = 6\sigma + 1 = 31$. Both kernels were normalized so $\sum w = 1$.

Figure 10: Comparison of Gaussian filtering: Naive 2D vs. Separable method.

Measured runtimes:

Method	Time (s)
Naive 2D Gaussian	4.668
Separable Gaussian	6.648

Table 1: Runtime comparison of Gaussian filtering methods.

Observation: Theoretically, separable convolution is faster since complexity reduces from $O(k^2N)$ to $O(2kN)$ for image size N. However, our Python loop implementation incurs large overhead, and separable requires two full passes. In optimized C/C++ libraries such as OpenCV, separability indeed yields significant speed-ups.

4 Task 5(c): Order of Convolution (Gaussian and Laplacian)

We compared three mathematically equivalent operations:

$$l * (g * f), \quad g * (l * f), \quad (l * g) * f$$

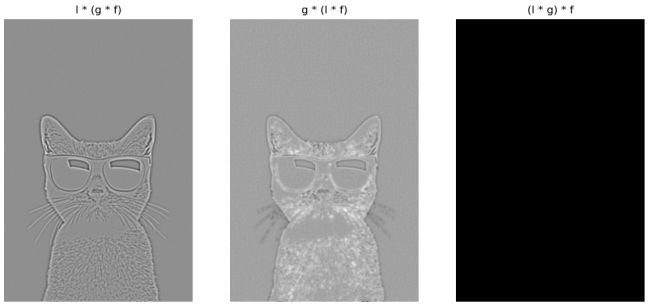


Figure 11: Comparison of Laplacian and Gaussian order of convolution.

Discussion

In theory, the three methods should yield identical results since convolution is commutative and associative. In practice, differences arise due to:

- **Finite precision:** Floating-point rounding and clipping effects differ depending on order.
- **Kernel truncation:** The Gaussian kernel is truncated to finite size.
- **Border effects:** Padding interacts differently with filtering order.
- **Implementation caveat:** Direct convolution of two kernels with an image-style function can zero out values if cast to 8-bit. Correct computation of $l * g$ should keep the kernel in floating-point.

Part 6: Intensity Transformation and Convolution

5 Task 6(a): Linear Intensity Transformations (Mathematical Analysis)

Let the pointwise intensity transformation be

$$T(r) = ar + b,$$

with $a, b \in R$, and let w be a fixed convolution kernel. We ask if there exists a T' such that

$$T'(w * f) \;=\; w * T(f), \quad \forall f.$$

Derivation

Expanding,

$$T(f) = af + b\mathbf{1},$$

where $\mathbf{1}$ is the constant image of ones. By linearity,

$$w * T(f) = a(w * f) + b(w * \mathbf{1}).$$

For a constant image,

$$(w * \mathbf{1})(x) = \sum_u w(u) = S,$$

where S is the DC gain of the kernel. Thus,

$$w * T(f) = a(w * f) + bS.$$

Hence a suitable T' is

$$\boxed{T'(r) = ar + bS, \quad S = \sum_u w(u).}$$

Special Case: Normalized Kernel

If $\sum w(u) = 1$, then $T'(r) = ar + b$, so $T' = T$ and the operations commute exactly.

When does this fail?

- **Edge renormalization:** If kernels are renormalized near borders, S becomes position-dependent.
- **Adaptive kernels:** If w changes with position or data, no single T' works.

6 Task 6(b): Nonlinear Intensity Transformations

For nonlinear T, such as gamma correction, the order matters. Gamma correction was defined as

$$I_{\text{out}} = 255 \cdot \left(\frac{I_{\text{in}}}{255}\right)^{\gamma},$$

computed in float and clipped to $[0, 255]$.

We tested $\gamma = 0.2$ with a disk kernel ($r = 6$, normalized).

Figure 12: Grayscale image: gamma correction before vs after convolution.

Discussion

- **Gamma before convolution:** Expands dark regions, then blur spreads the contrast changes.
- **Gamma after convolution:** Matches what an out-of-focus camera would capture (blur first, then tone-mapping).

7 Color Images in HSV and Lab

We repeated the test in HSV and Lab luminance spaces.

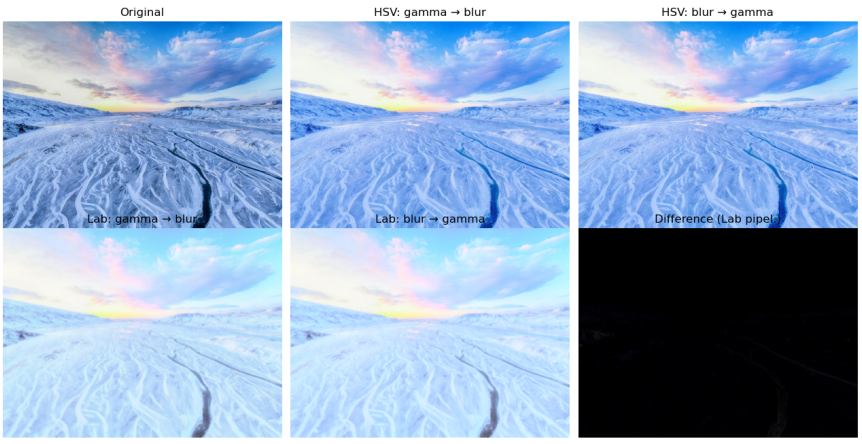


Figure 13: Gamma before vs after convolution in HSV and Lab color spaces.

8 Timing Analysis

Timings with radius $r = 6$:

Method	Time (s)
HSV: Gamma before convolution	34.1790
HSV: Gamma after convolution	31.4169
Lab: Gamma before convolution	31.5508
Lab: Gamma after convolution	13.9886

Table 2: Runtime results for gamma correction before vs after convolution.

Figure 14: Timing visualization for HSV and Lab pipelines.

Discussion

- Lab conversions are heavier.
- Gamma-after is consistently faster because fewer high-contrast regions are blurred.
- Gamma-before produces visually distinct spreading of shadows.

Part 7: Colour Space Transformations and Object Recoloring

9 Task 7(a): HSI Conversion

The RGB image was converted to HSI, and the channels visualized.

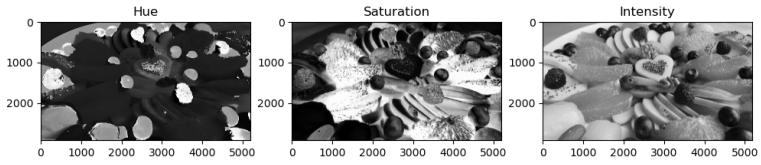


Figure 15: HSI decomposition: Hue, Saturation, Intensity channels.

10 Task 7(b): Colour Slicing

We picked seed $(100, 100)$ with thresholds $\Delta H = 0.05,\ \Delta S = 0.30,\ \Delta I = 0.30$. The resulting HSI mask is:

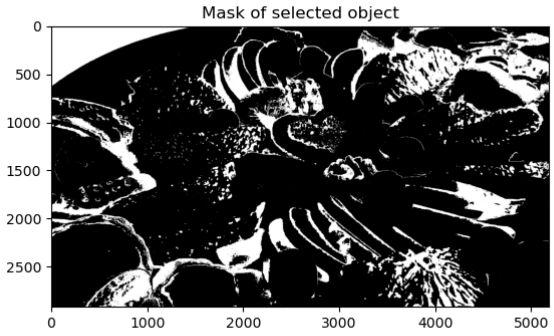


Figure 16: Binary mask of selected object in HSI.

We also performed slicing in RGB with cuboid thresholds (20,80,80) around the seed RGB. The comparison shows HSI produces cleaner segmentation while RGB leaks more background.

11 Task 7(c): Colour Transformation

The object was recoloured to target $c_t = (H, S, I) = (0.8, 0.9, 0.7)$. We used additive shift for Hue and Intensity and multiplicative scaling for Saturation.

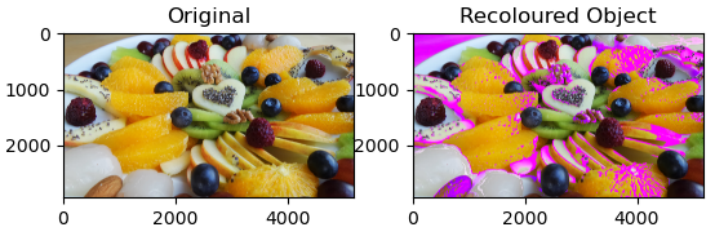


Figure 17: Original vs recoloured object after transformation.

Discussion

- Additive transformations fit Hue (circular) and Intensity (brightness shift).
- Multiplicative scaling is natural for Saturation, preserving neutral gray at $S = 0$.
- Mask leakage explains partial recolouring of the background.

Conclusion

We implemented convolution and Laplacian/Gaussian filters, compared naive and separable Gaussian convolution, studied Laplacian-of-Gaussian orders, and explored intensity transformations. We demonstrated that linear T can commute with convolution (with modified bias), but nonlinear T cannot. Gamma-after convolution resembles physical imaging. Finally, HSI colour slicing and transformation effectively recoloured objects, showing the advantage of perceptually meaningful colour spaces.