IN4640 Assignment 1 on Intensity Transformations and Neighborhood Filtering

Ranga Rodrigo

January 15, 2026

1. Apply the following intensity transform to the image in Fig. 1:

Figure 1: Runway image for 1.

   (a) Gamma correction with $\gamma = 0.5$.

   (b) Gamma correction with $\gamma = 2$.

   (c) Contrast Stretching (linear piecewise transformation)

$$s(r) = \begin{cases} 0, & r < r_1, \\ \dfrac{r - r_1}{r_2 - r_1}, & r_1 \le r \le r_2, \\ 1, & r > r_2, \end{cases}$$

   where
   r : input pixel intensity, normalized to the range $[0, 1]$.
   $s(r)$: output pixel intensity after contrast stretching.
   $r_1 = 0.2, \qquad r_2 = 0.8$

2. Consider the image shown in Fig. 2[1].

   (a) Apply gamma correction to the L plane in the $L^*a^*b^*$ color space and state the γ value.

   (b) Show the histograms of the original and corrected images.

3. Write your own function to equalize the histogram of an image. Apply this function to the runway image.

Figure 2: Image for gamma correction.

Figure 3: Woman standing in front of an open door.

4. Fig. 3 is a photo of a woman standing in front of an open window[2]. Convert this to grayscale.

   (a) Use Otsu thesholding to obtain the binary mask for the foreground comprising the woman and the room. Report the resulting threshold value.

   (b) Carry out histogram equalization only for the foreground region. What are the hidden features that are revealed in the resulting image?

5. Gaussian filtering:

   (a) Using NumPy, compute a normalized 5×5 Gaussian kernel for $\sigma = 2$.

   (b) Visualize a 51×51 computed Gaussian kernel as a 3D surface plot, where the kernel coefficients represent the height.

   (c) Apply Gaussian smoothing to a given grayscale image using the manually computed Gaussian kernel.

   (d) Do the same using OpenCV's built-in `cv.GaussianBlur()` function.

6. Derivative of Gaussian:

   (a) Consider the two–dimensional Gaussian function

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right).$$

[1] https://www.adobe.com/creativecloud/photography/discover/highlights-and-shadows.html

[2] Ronak Valobobhai, https://unsplash.com/photos/a-woman-standing-in-front-of-an-open-door-6YzA45_b2vA.

Show that its first–order partial derivatives are given by

$$\frac{\partial G}{\partial x} = -\frac{x}{\sigma^2} G(x,y), \qquad \frac{\partial G}{\partial y} = -\frac{y}{\sigma^2} G(x,y).$$

(b) Using NumPy, compute normalized 5×5 kernels corresponding to the derivatives of a Gaussian for $\sigma = 2$ in the x- and y-directions.

(c) Visualize a 51×51 derivative-of-Gaussian kernel (for either the x or y direction) as a 3D surface plot, where the kernel coefficients represent the height.

(d) Apply the computed derivative-of-Gaussian kernels to a given grayscale image to obtain the image gradients in the horizontal and vertical directions.

(e) Using OpenCV, compute the image gradients by applying `cv.Sobel()`. Compare the results with those obtained above and comment on any observed differences.

7. Write a program to zoom images by a given factor $s \in (0, 10]$. You must use a function to zoom the image, which can handle

   (a) nearest-neighbor, and

   (b) bilinear interpolation.

   I have included several images, large originals, and their zoomed-out versions. Test you algorithm by computing the normalized sum of squared difference (SSD) when you scale-up the given small images to match the size of the large original images. The SSD should be small when comparing with the original images.

8. Fig. 4 is an image corrupted with salt and pepper noise.

   (a) Apply Gaussian smoothing.

   (b) Apply median filtering.

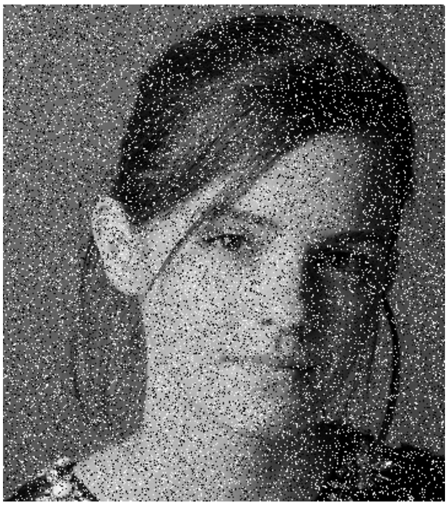

Figure 4: Image corrupted with salt and pepper noise.

9. Carry out image sharpening on am image of your choice.

10. Bilateral filtering:

    (a) Write a Python function to manually implement a bilateral filter for grayscale images. Take as input a grayscale image, a kernel diameter, a spatial standard deviation σ_s, and a range (intensity) standard deviation σ_r

(b) Apply Gaussian smoothing using OpenCV's `cv.GaussianBlur()` function.

(c) Bilateral filtering using OpenCV's `cv.bilateralFilter()` function.

(d) Your manually implemented bilateral filter from part (a).

GitHub Profile

You must include the link to your GitHub (or some other SVN) profile, so that I can see that you have worked on this assignment over a reasonable duration. Therefore, make commits regularly. However, I will use only the pdf for grading to save time.

Submission

Upload a report (eight pages or less) named as your_index_a01.pdf. Include the index number and the name *within the pdf* as well. The report must include important parts of code, image results, and comparison of results. The interpretation of results and the discussion are important in the report. Extra-page penalty is 20 marks per page.