CS 663: Digital Image Processing

Assignment 1

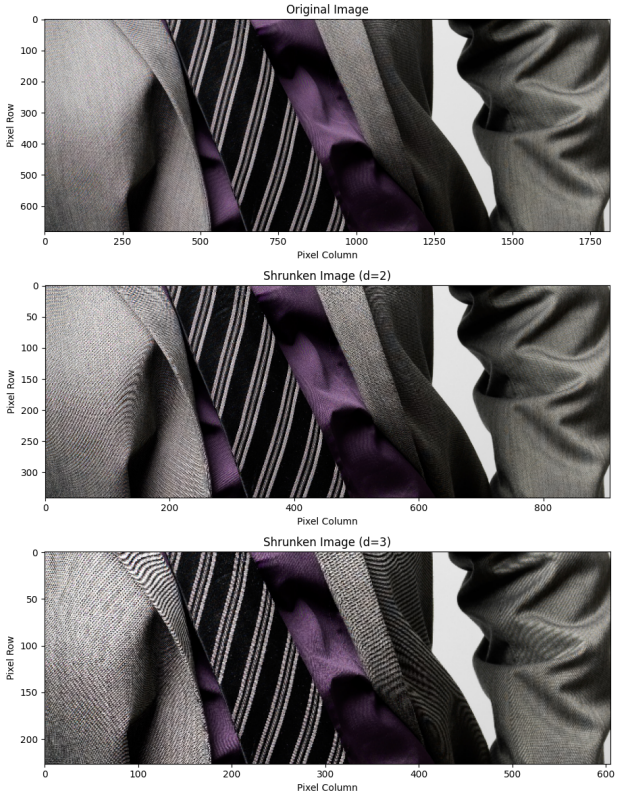
Pratiek Sonare (22B2440)
Entenuka Jogarao (22B2702)
Instructor: Suyash P. Awate

August 29, 2025

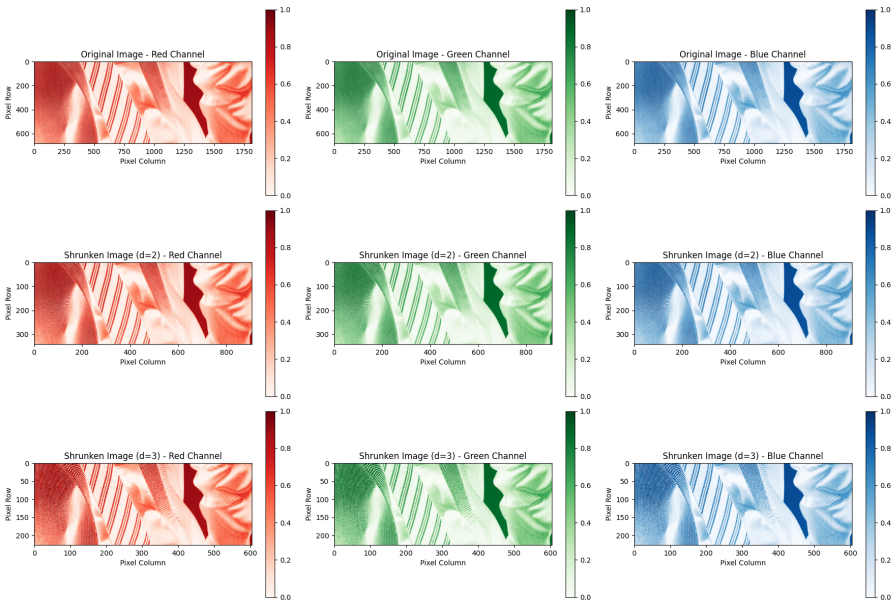
1 Image Subsampling and Interpolation

1.1 Image Shrinking

Function: `myImageShrink()`
Summary: We shrink the image by skipping every d-th sample, effectively sub-sampling the image. This results in Moire effect - where we see a distorted image with black pixels Distortions are clearly visible by zooming in the images shrunk by d=2 and d=3



(a) Original and Shrinked Images (with Moire Effect)



(b) RGB channels for each subsampled image

Figure 1: Image Shrinking results.

1.2 Nearest Neighbor Interpolation

Function: `myNearestNeighborInterpolation()`
Summary: Enlarging the image to { 300(M-1) + 1, 300(N-1) + 1 } from { M, N } pixel range using Nearest Neighbour interpolation — each new pixel takes the value of the nearest pixel from the original image. The original image was in gray-scale, and the intensities are colour coded using the `jet` colormap.

1.3 Bilinear Interpolation

Function: `myBilinearInterpolation()`
Summary: Given a pixel location (x, y) in the resized image, mapped to fractional coordinates in the original image, let If the intensities of the four neighboring pixels are

$$f(x_1, y_1), \quad f(x_2, y_1), \quad f(x_1, y_2), \quad f(x_2, y_2),$$

then the interpolated value at (x, y) is computed as

$$f(x, y) = f(x_1, y_1)(1-dx)(1-dy) + f(x_2, y_1)(dx)(1-dy) + f(x_1, y_2)(1-dx)(dy) + f(x_2, y_2)(dx)(dy).$$

1.4 Bicubic Interpolation

Function: `myBicubicInterpolation()`
Summary: Bicubic interpolation considers a 4×4 neighborhood of pixels around $(x_{\text{int}}, y_{\text{int}})$. The interpolation value is computed as $f(x, y) = \sum_{m=-1}^{2} \sum_{n=-1}^{2} f(x_{\text{int}}+m,\ y_{\text{int}}+n)\, w(m-dx)\, w(n-dy)$, where $w(\cdot)$ is the cubic kernel.

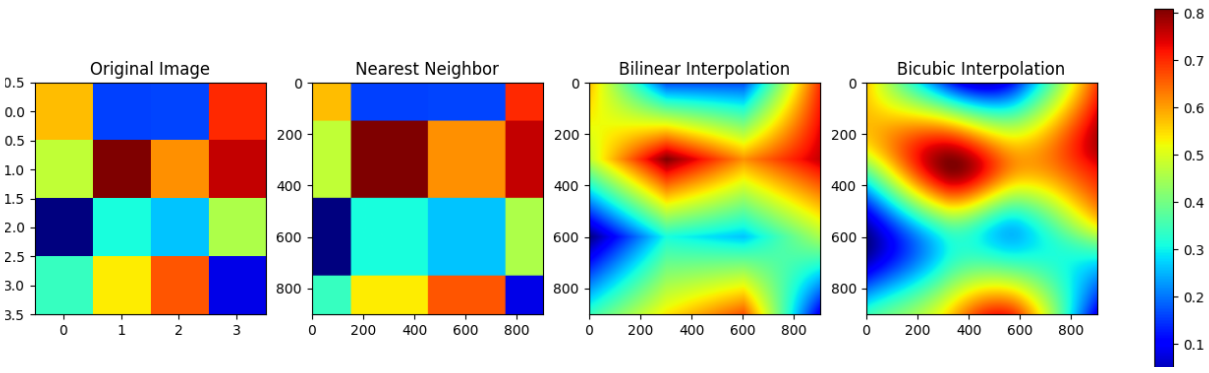


Figure 2: Image enlargement using NN, Bilinear, Bicubic interpolation

1.5 Image Rotation

Function: `myImageRotationUsingBilinearInterp()`
Function: `myImageRotationUsingNearestNeighborInterp()`

Summary: We calculate the rotation matrix for an input angle (here, 5 deg) and find the dot product with the image matrix to find the rotated image coordinates. Since, these coordinates may not always correspond to integer values, we interpolate them using NN / Bilinear interpolation methods to achieve the final image. Notice how empty spaces are blacked out after rotation!

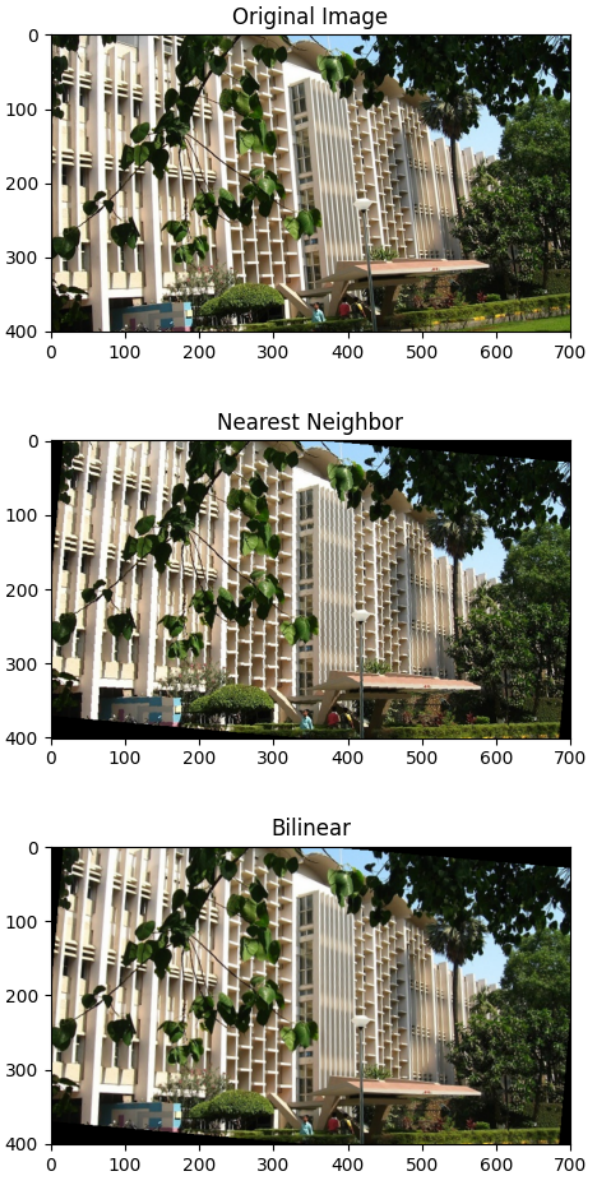


Figure 3: Image rotation using NN, bilinear interpolation.