1.6 Downsampling and Upsampling

Summary: To upsample or enlarge the subsampled CT scan images, we use and compare NN, Bilinear and Bicubic interpolation methods.
Key Observations: We find the NN method has the highest RMSE, also visible in the difference plot, while we see little to no difference in the bilinear and bicubic plots.

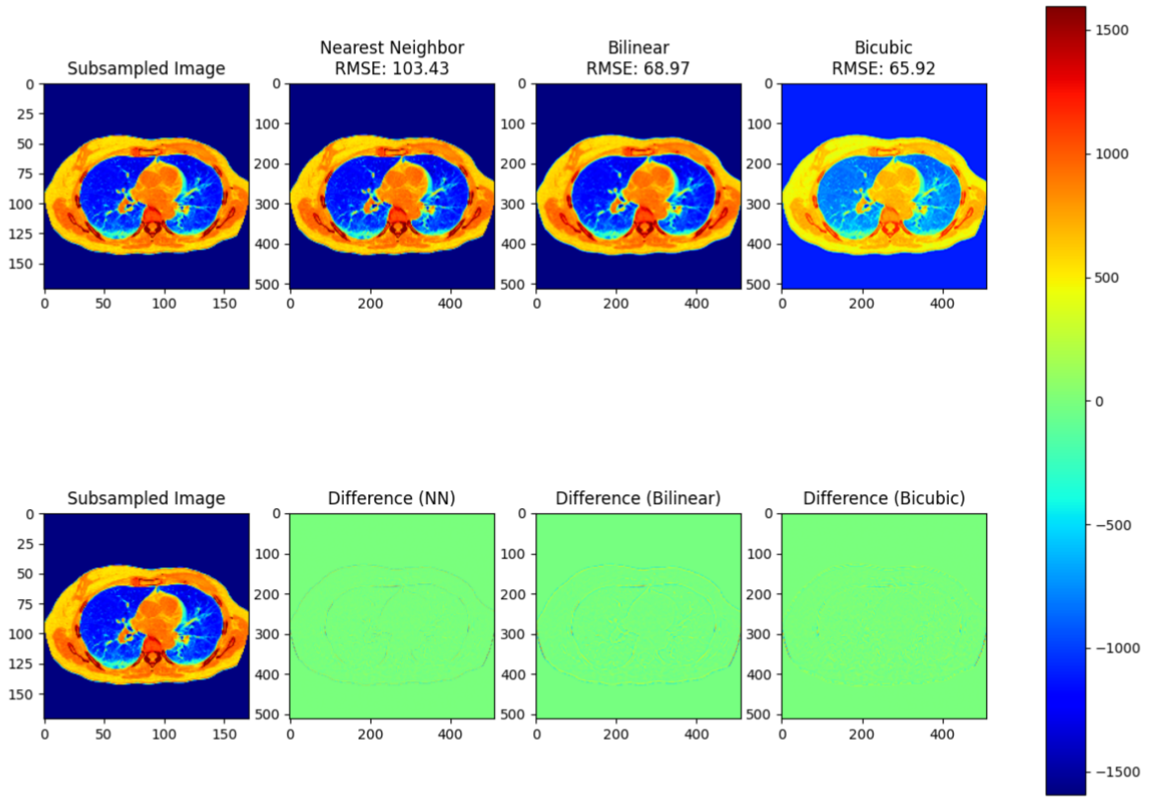

Figure 1: Chest CT Scans interpolation with difference using NN, Bilinear and Bicubic

Conclusion: For this Chest CT Scan, Bilinear and Bicubic interpolation methods work best since they result in the lowest RMSE and visible difference!

2 Thresholding

Input Images: The input images are in the .png and .tif format. All these images have varying maximum pixel intensities. Libraries such as `scikit-image` expect pixel intensities in the range of [0-255] (uint8 format). Hence, the intensities were scaled appropriately.

2.1 Manual Thresholding

Function: `myManualThreshold()`
Summary: By taking the average of RGB channels, we first convert the input image to grayscale and then input a threshold of 175, where intensity at a given pixel if ¡ 175, then it is assigned 0 (black) otherwise 255 (white)

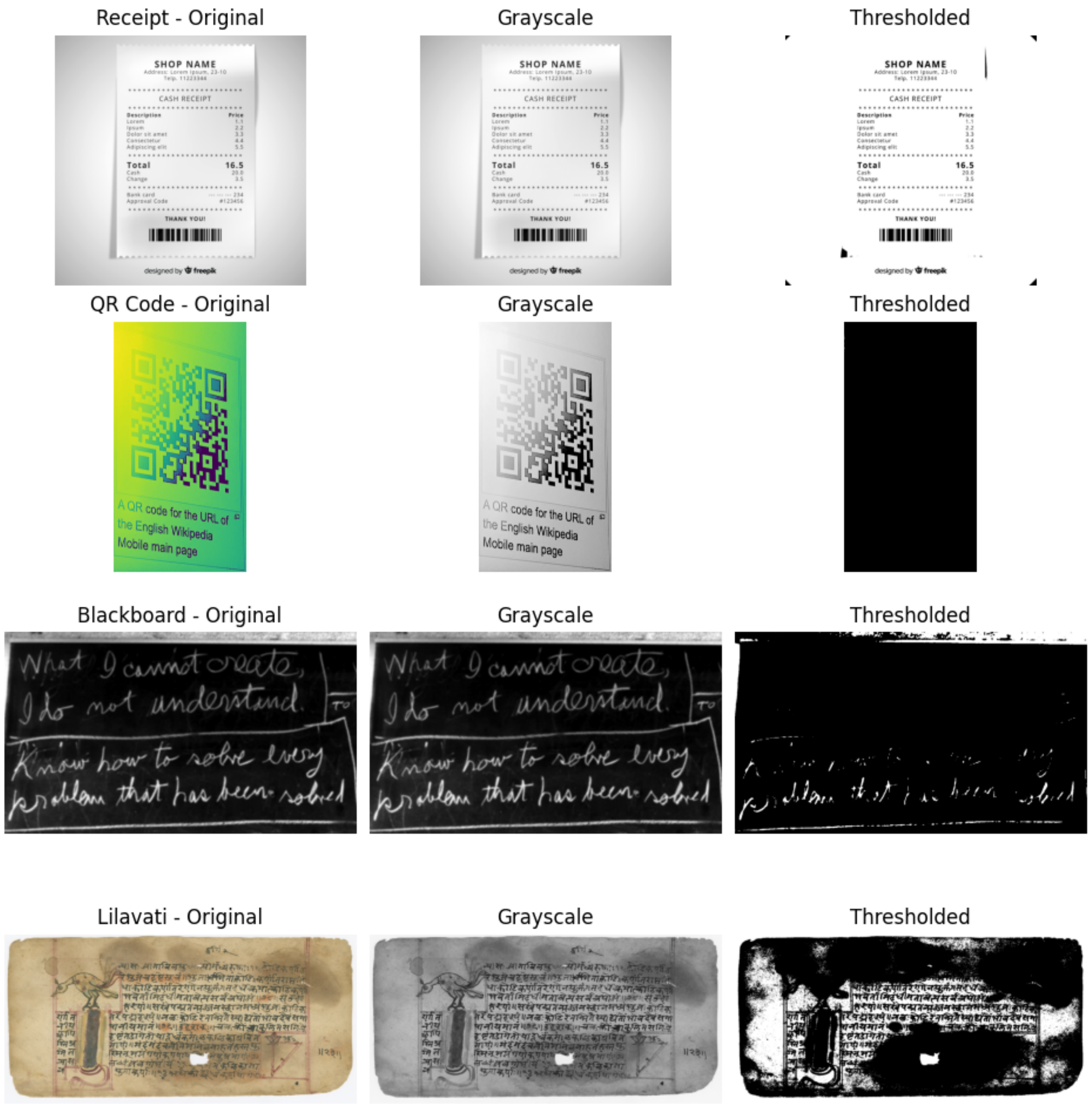

Figure 2: Manual thresholding results

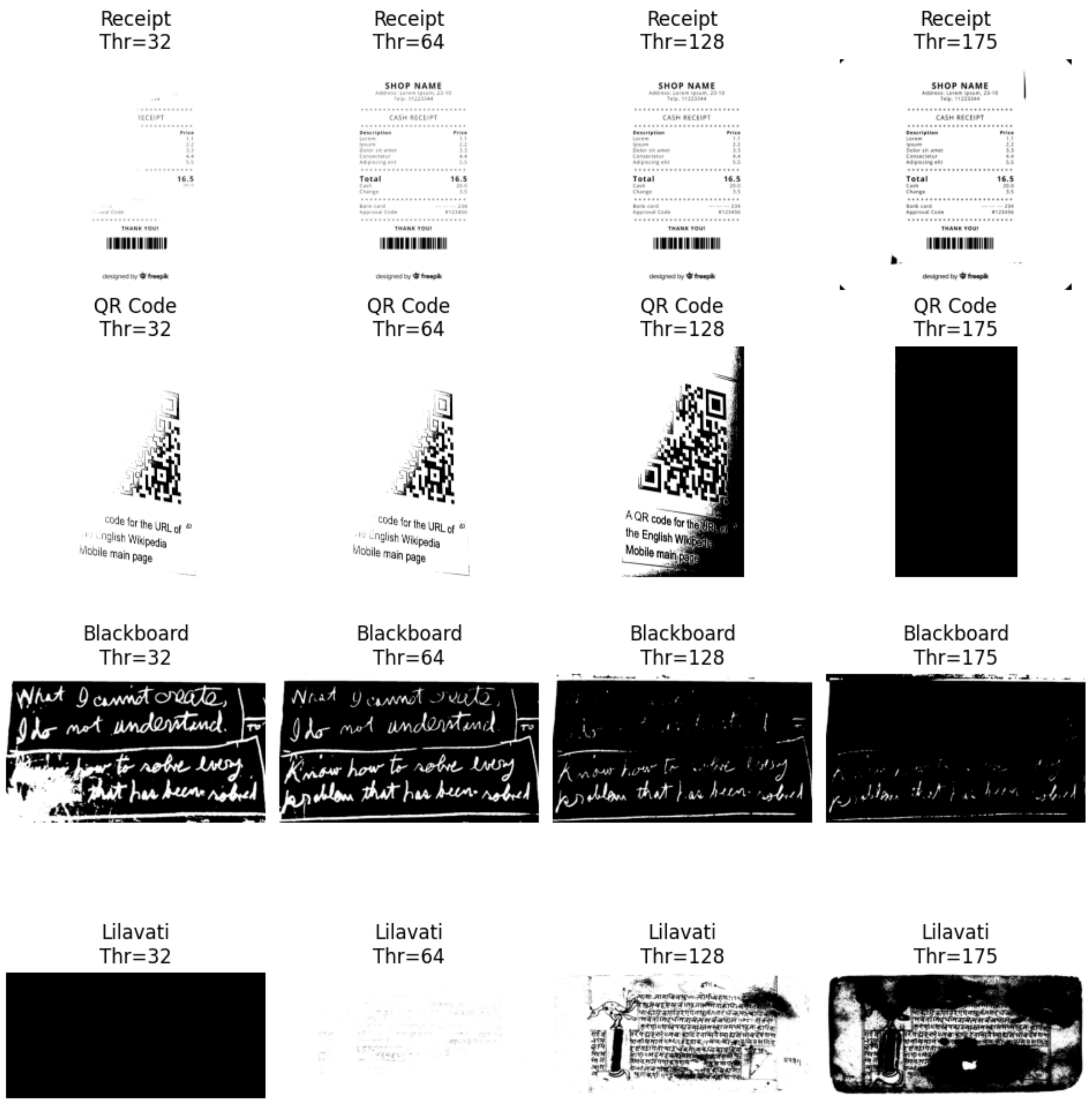

Figure 3: Comparing images for different thresholds

Key Observations: Thr=75 works best for the receipt, it may not work the best for the rest of the images where the QR or written text is not clearly visible, meaning even if they are written in white, the pixel intensity < 175 hence giving a black color! However, for lighter shades of shalk, they are classified as white when you bring down the threshold to Thr $= 32$

Conclusions: Lower threshold values can correctly classify lighter shades of white, however undermining black colors at the same time (notice Lilavati.tif) and vice-versa.

2.2 Otsu Thresholding

Function: `myOtsuThreshold()`
Summary: Otsu's method is an automatic image binarization technique that determines an optimal threshold by analyzing the image histogram and selecting the value that *minimizes the within-class variance* (or equivalently, *maximizes the between-class variance*) between foreground and background. This makes it adaptive to the image's intensity distribution, eliminating the need for a manually chosen threshold.
Key Observations: We observe better results for all four input images. Another key observation is that, all images have different Otsu Thresholds.

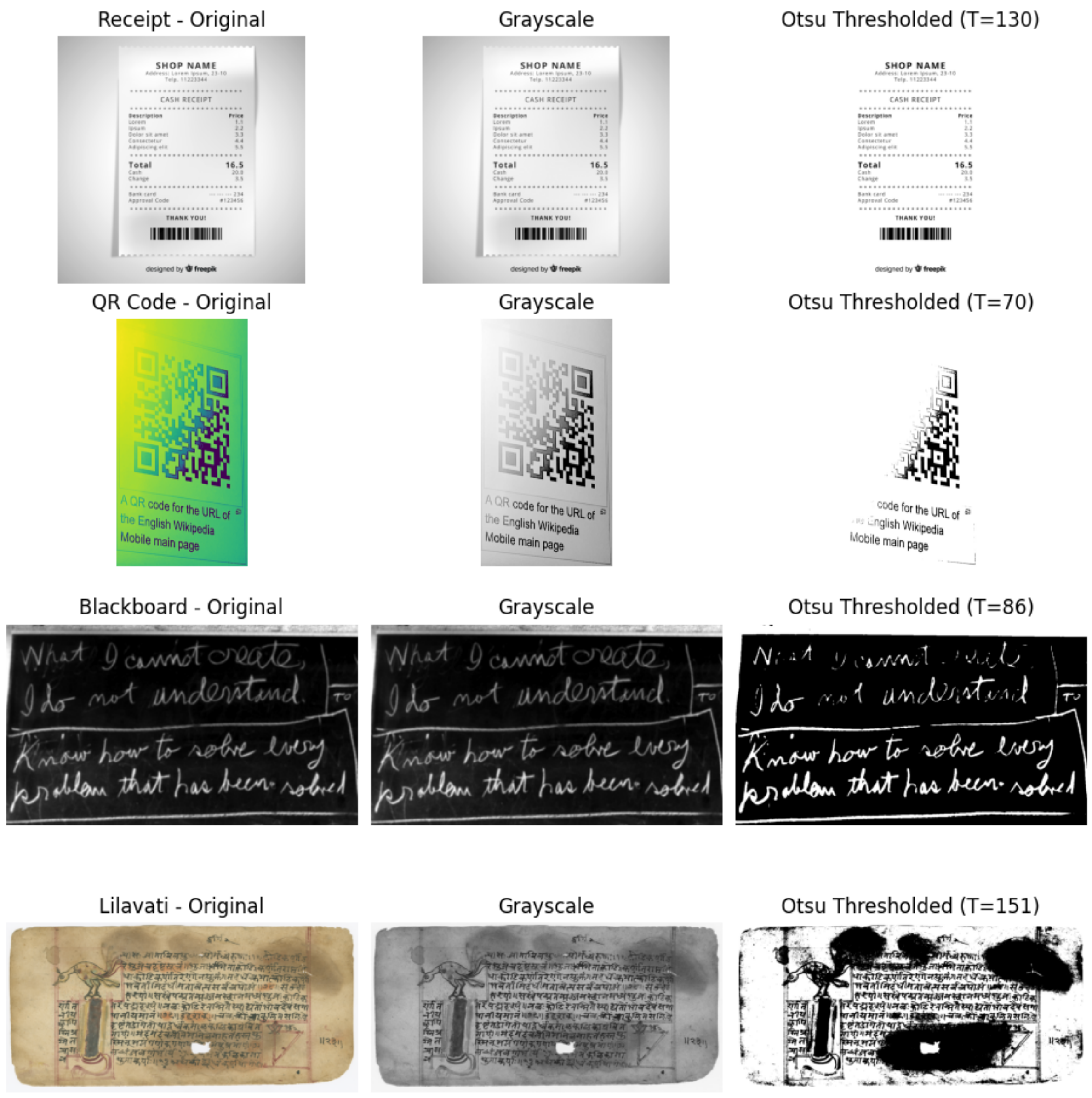

Figure 4: Otsu Threholded Images

2.3 Local/Adaptive Thresholding

Function: `myAdaptiveThreshold()`
Summary: We use Niblack's Adaptive Thresholdin method here which is a *local adaptive thresholding* technique used to binarize images with varying illumination or background. Instead of a single global threshold, the threshold at each pixel is computed based on the local mean and standard deviation in a sliding window centered at that pixel. The threshold is defined as:

$$T(x, y) = m(x, y) + k \cdot \sigma(x, y)$$

where $m(x, y)$ and $\sigma(x, y)$ are the local mean and standard deviation of the neighborhood around pixel (x, y), and k is a user-defined parameter (typically in the range -0.5 to -0.2).

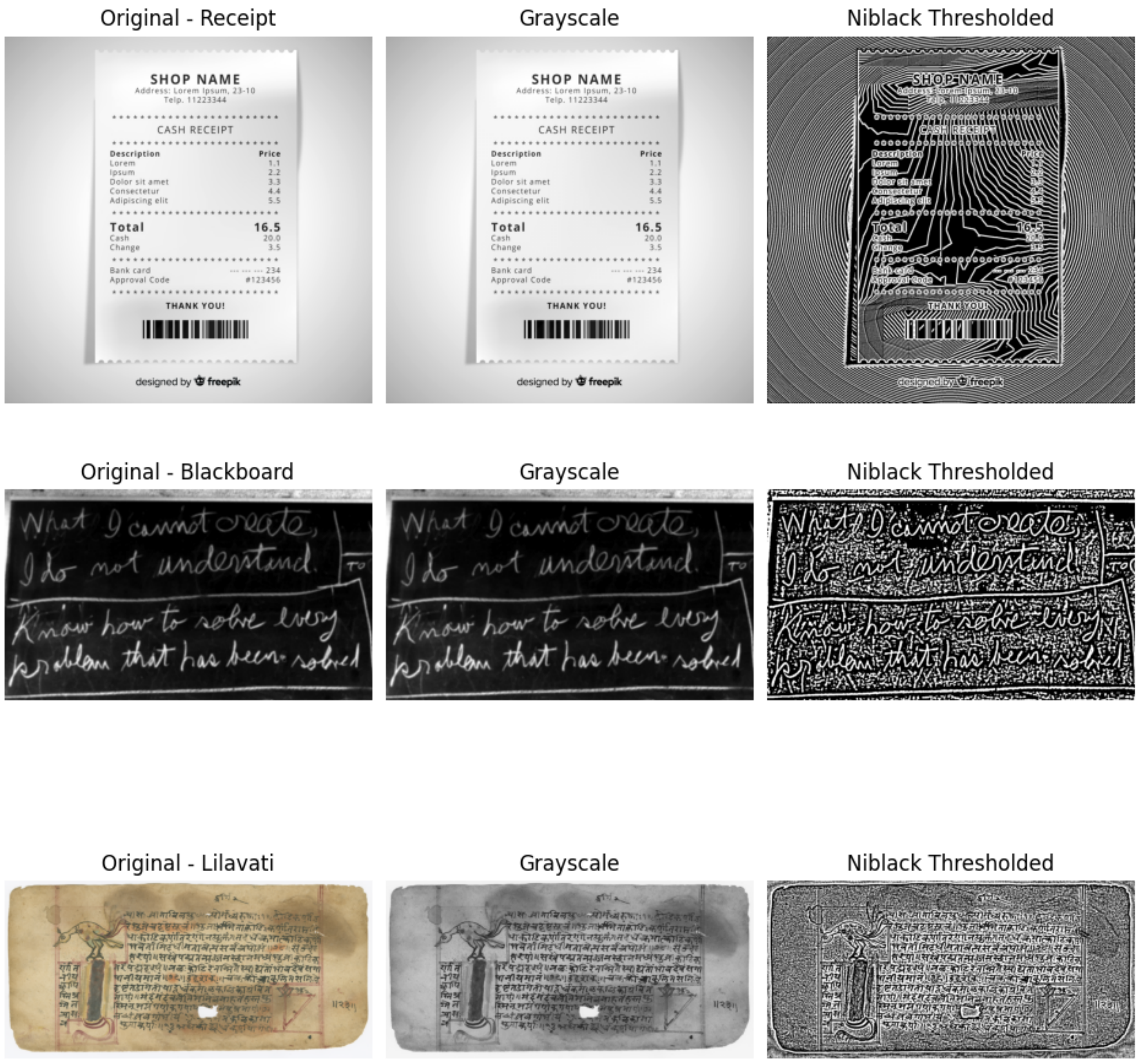

Figure 5: Niblack's Adaptive Thresholding

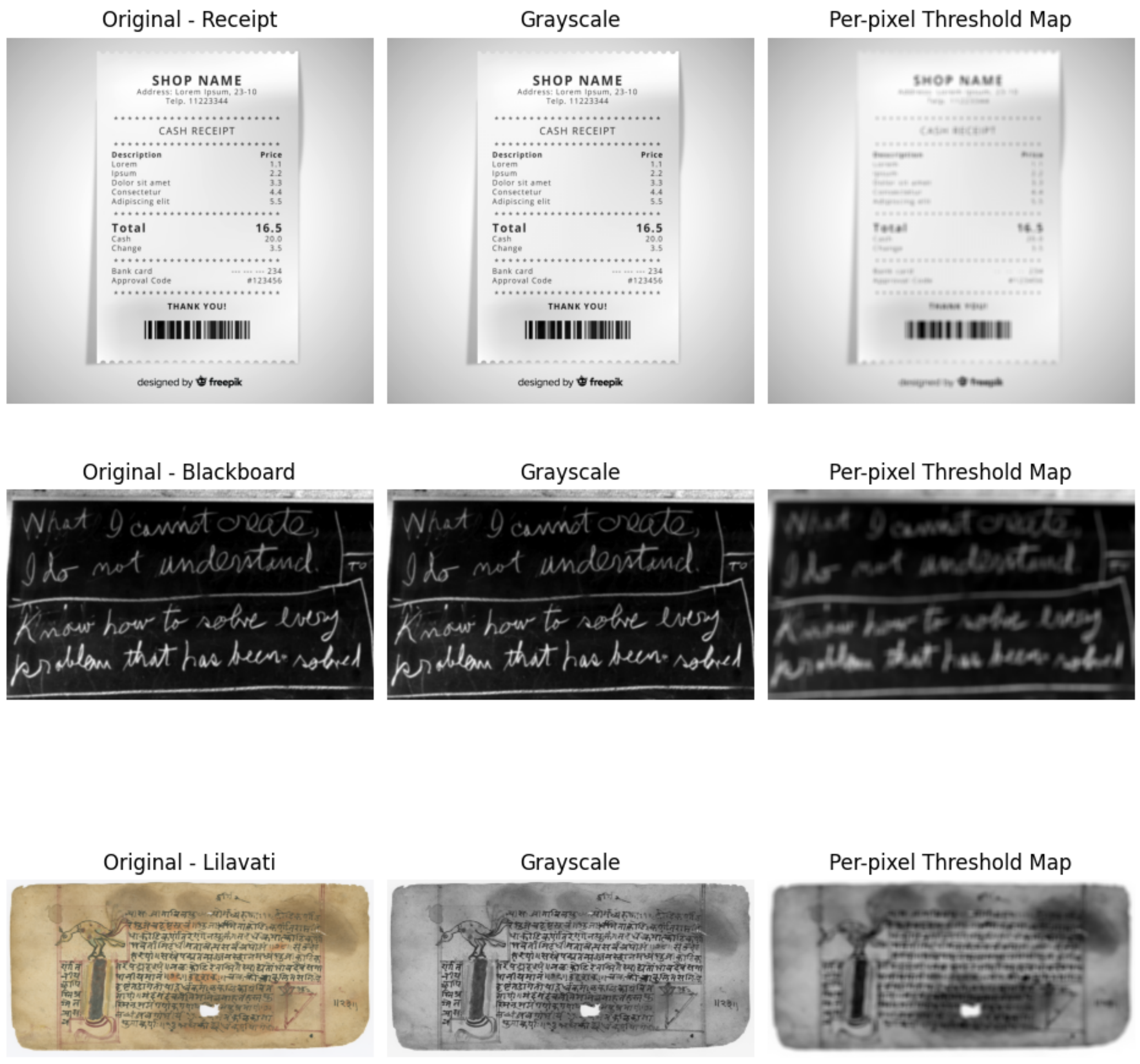

Figure 6: Perpixel threshold values of adaptive thresolding

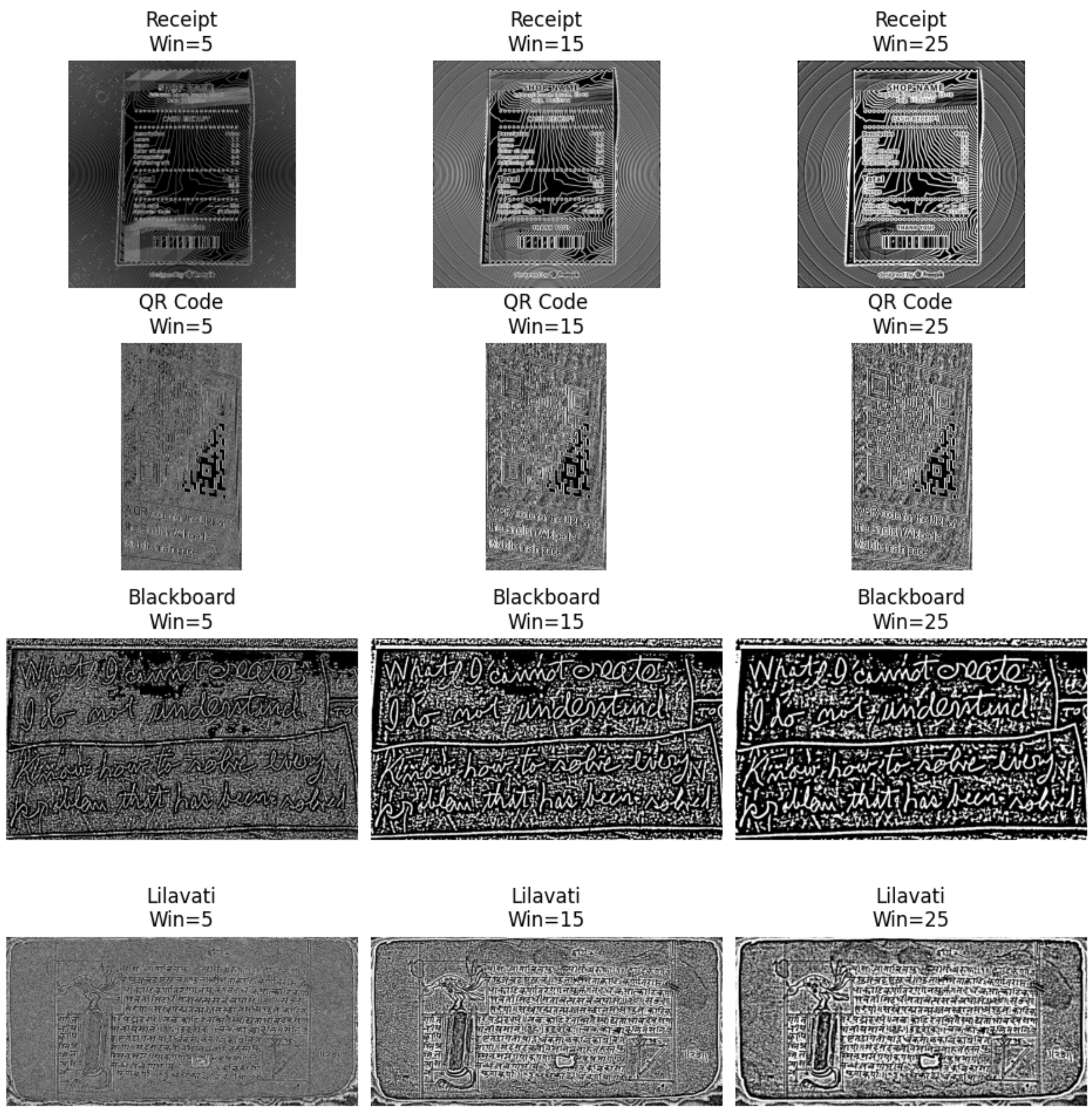

Figure 7: Comparing images for different thresholds