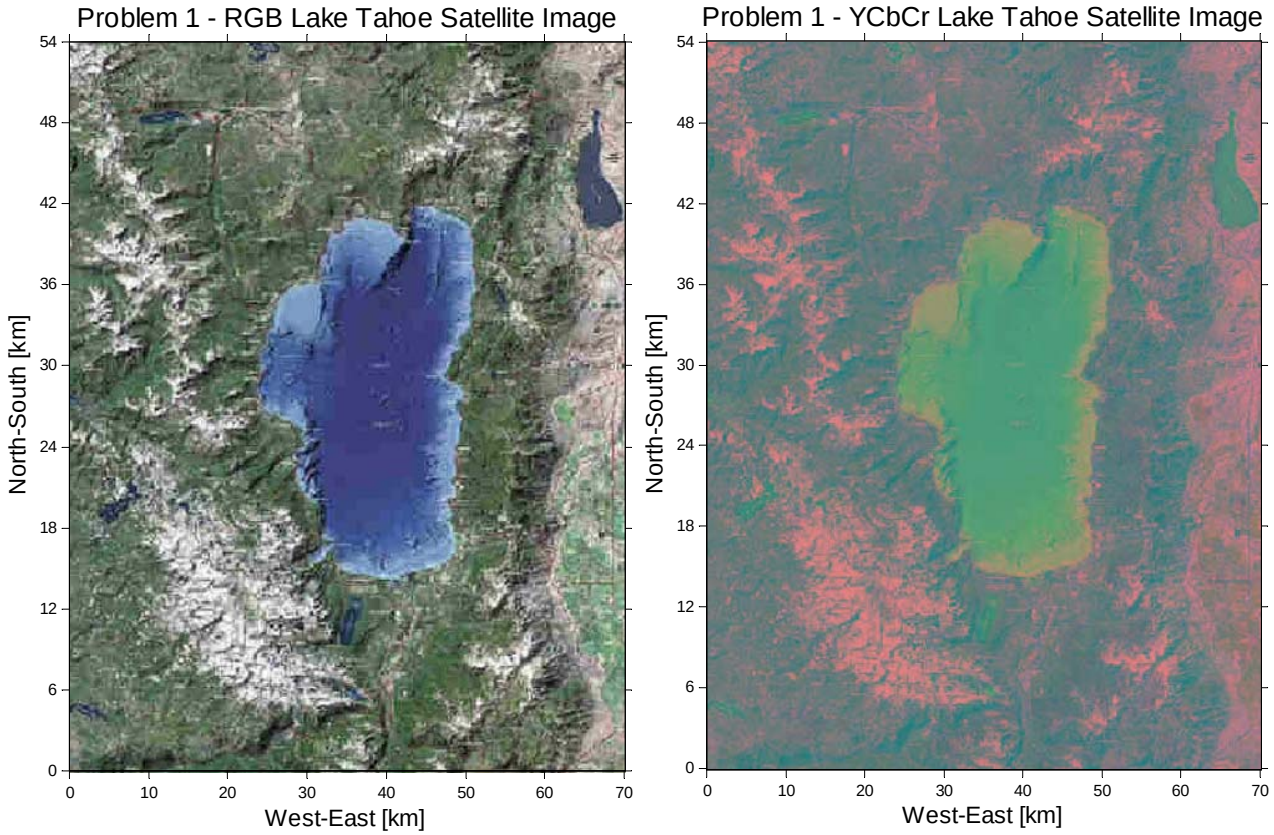
Christopher Tsai
April 22, 2007
EE 368 – Digital Image Processing

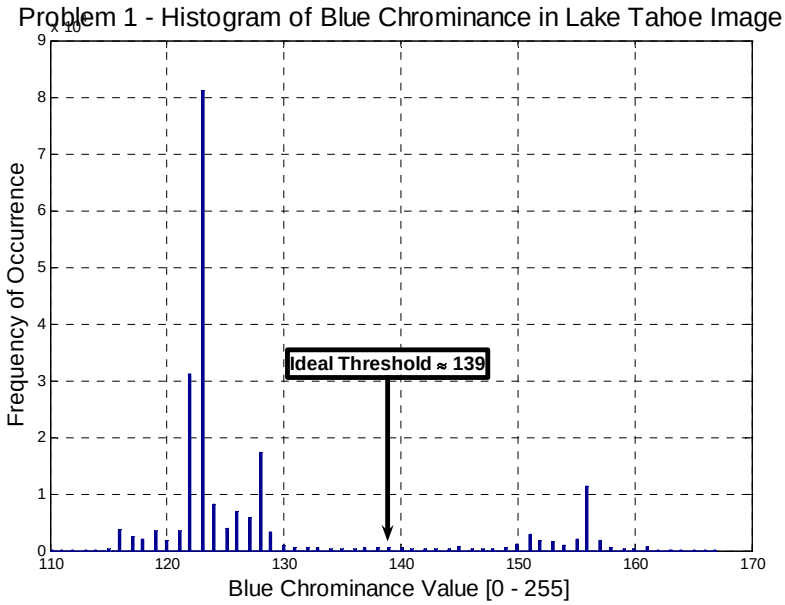
Problem Set II

Problem 1 – Satellite Image Processing

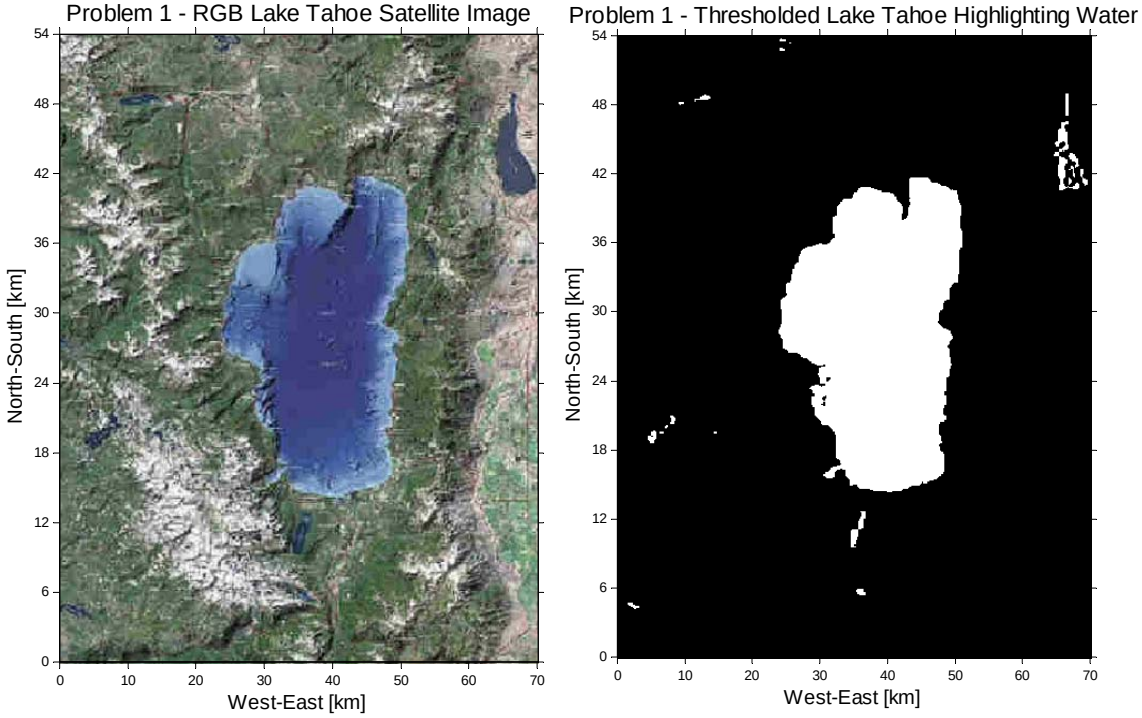
Because Lake Tahoe fails to dominate any single component of the primary Red-Green-Blue (RGB) color space, we first convert our image to the Luminance-Chrominance (YCbCr) color space. As a YCbCr image, Lake Tahoe protrudes readily from its surroundings in the blue chrominance (Cb) channel, where bodies of water appear with an especially distinct signature, as seen on the right:



In order to determine an appropriate threshold to isolate water from land, we iteratively compute the midpoint between the two threshold regions until this midpoint converges. Beginning with a threshold estimate of 255/2 = 127.5, we segment the blue chrominance (Cb) image into two areas, each with its own average intensity; advancing our estimate to the midpoint between these two intensity values, we produce another segmentation, and we continue in this fashion until the midpoint ceases to change, yielding a threshold value that rounds to 139. The Cb image histogram validates our choice:

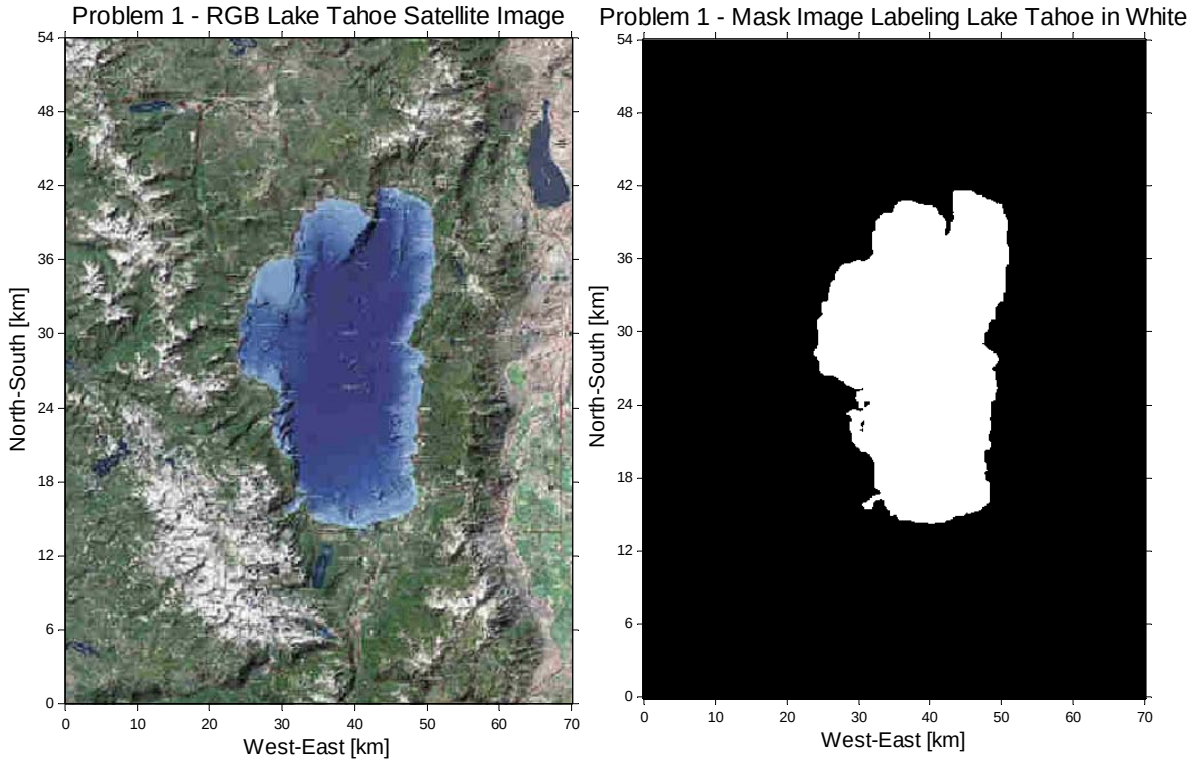


As we observe from the histogram, the point of convergence (139) lies approximately halfway between the less-intense land background (left distribution) and the strong water signature (right distribution). Using this threshold, we obtain a binary image highlighting all watery locations:



Since we seek to characterize only the main central lake, we must eliminate all of the smaller regions from our binary mask. After employing the Matlab command `bwlabel` to numerically tag all of the eight-

connected regions, we determine the numbered region with the most pixels in our thresholded image by computing the non-zero mode; in order to compute this statistical mode, we must first rarify the zero-labeled background pixels by random noise (`randn`), hence making the main lake body the most dominant region. Upon setting all non-mode values to zero, we obtain the main lake binary mask:



By counting the white pixels in the binary mask, we conclude that the central lake in our digital image comprises **28,646 pixels**.

Furthermore, if we know that the height and width of the image correspond to 69.01 km and 52.58 km, respectively, then we can approximate the surface area of Lake Tahoe by multiplying the pixel spacing by the previously computed pixel count:

$$\textbf{Surface Area of Lake Tahoe} \approx \frac{69.01\text{ km}}{525\text{ pixels}} \cdot \frac{52.58\text{ km}}{400\text{ pixels}} \cdot 28{,}646\text{ pixels}^2 \approx \boxed{\mathbf{494.968205\text{ km}^2.}}$$

Problem 2 – Image Magnification through Interpolation

Please refer to the hand-written section for solutions to Problem 2.1 and Problem 2.2.

In Problem 2.3, we examine the nearest-neighbor and bilinear interpolation algorithms when applied to (256 bit × 256 bit) images magnified by factors of m = 1.1 and m = 1.3.

We begin by magnifying the cameraman digital image using nearest-neighbor interpolation:

Problem 2.3B - Unmagnified Cameraman

Nearest-Neighbor Cameraman (m = 1.1)

Problem 2.3B - Unmagnified Cameraman

Nearest-Neighbor Cameraman (m = 1.3)

As we observe in the sleeve of the magnified cameraman images, nearest-neighbor interpolation generates artifacts along edges; the jaggedness of the cameraman's hair and jacket – especially for the larger magnification ($m = 1.3$) – tarnishes the subjective quality of the original image, since these boundaries should be straight lines, as the unmagnified image reveals. While these artifacts do not seem so severe for the smaller magnification, the larger magnification makes even the camera tripod appear splintered; in particular, notice how the white stripe on the left tripod leg now zigzags like a lightning bolt. Let us juxtapose these magnifications with the bilinearly interpolated images:

Problem 2.3B - Unmagnified Cameraman

Bilinear Interpolation Cameraman (m = 1.1)

We immediately notice a slight blur in the image, but the sleeve of the cameraman's jacket looks smoothly intact, without the artificial stair-steps produced in nearest-neighbor interpolation. We expect a certain degree of blurring because of the merging of four data values in the bilinear interpolation algorithm, but this merging appears to solve the problem of blocky edges, which arose primarily because nearest-neighbor interpolation simply chose the nearest neighbor without any regard for other nearby neighbors with intensely different pixel values. While the nearest-neighbor method was simpler to implement, the blind duplication of pixels created unnatural boundaries. The bilinear interpolation solves even the $m = 1.3$ case:

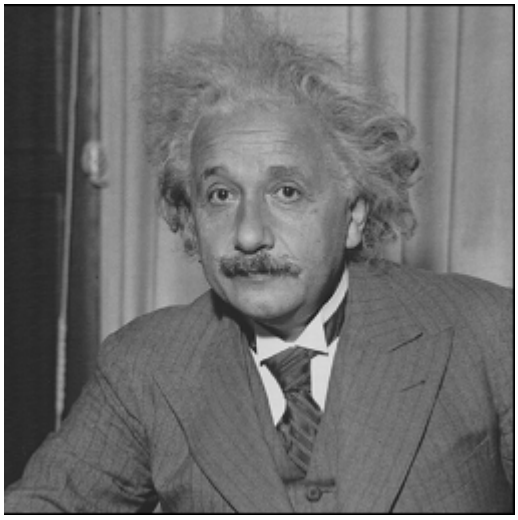
Problem 2.3B - Unmagnified Cameraman

Bilinear Interpolation Cameraman (m = 1.3)

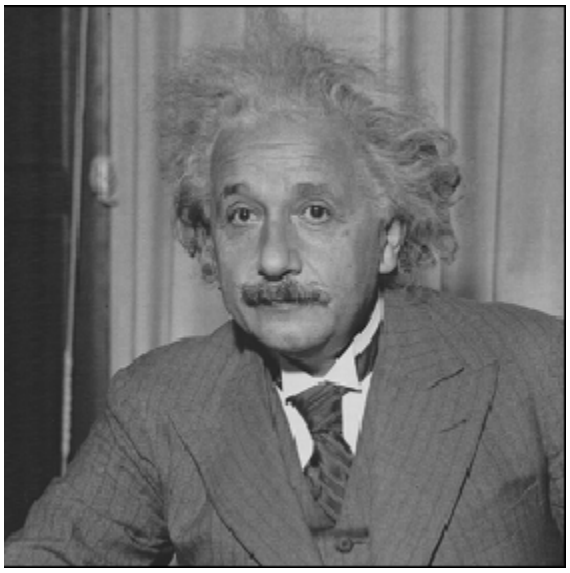
Juxtaposition beside the original image reveals only a slight blur, which we must expect from the addition of fabricated data points. However, most notably, the bilinearly interpolated magnifications contain no jagged edges, since each interpolated pixel value properly accounts for all surrounding pixel values; in other words, the added image data varies smoothly and continuously from pixel to pixel, unlike the pixel duplication used in nearest-neighbor interpolation. Besides the slight blur that image expansion has introduced, the bilinearly interpolated image exhibits no artifacts or eccentricities in either the small (m = 1.1) or large (m = 1.3) magnification. As a result, the subjective quality of the bilinear interpolation dwarves the jagged, visibly discontinuous image from nearest-neighbor interpolation.

We repeat our comparison exercise on the Einstein digital image with nearest-neighbor algorithm:

Problem 2.3B - Unmagnified Einstein



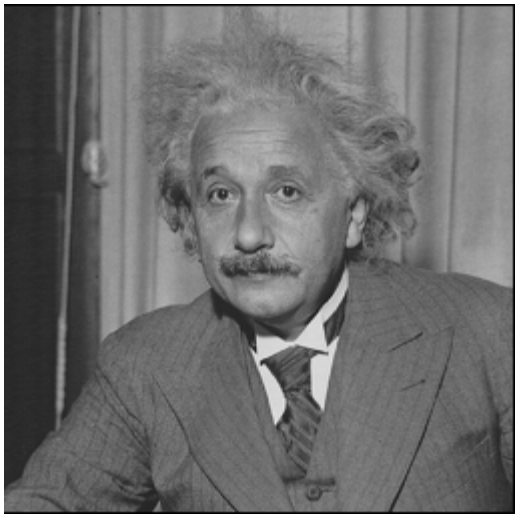
Nearest-Neighbor Einstein (m = 1.1)



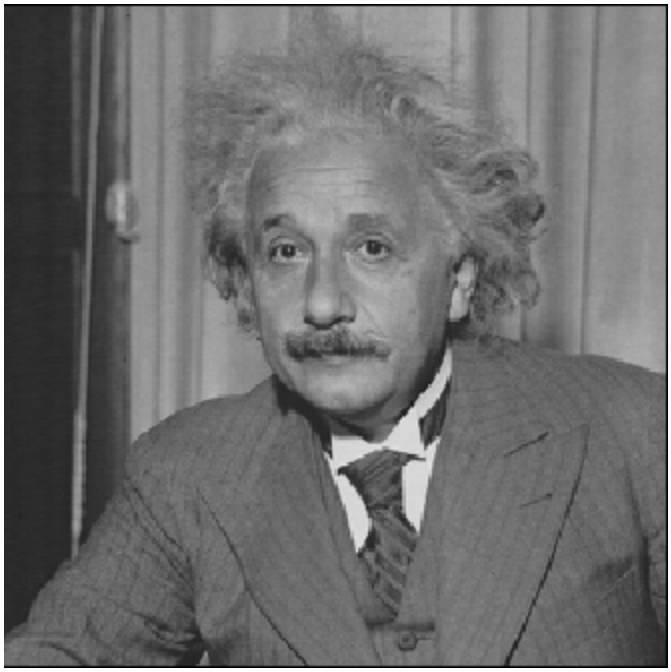
We first remark that, even more so than with the cameraman image, the Einstein image magnification with $m = 1.1$ does not flaunt noticeable flaw, perhaps because the image is too small for our eyes to notice slight imperfections along the edges. Moreover, the contrast between background and subject in the Einstein image pales in comparison to the contrast between the sky and cameraman's coat, making jagged edges more difficult to perceive in the Einstein image. The only visible artifact is the dark tie on Einstein's white shirt, which begins to appear slightly blocky because of nearest-neighbor blind duplication.

Let us examine the larger magnification, which, as expected, exposes the flaws of nearest-neighbor interpolation more blatantly:

Problem 2.3B - Unmagnified Einstein

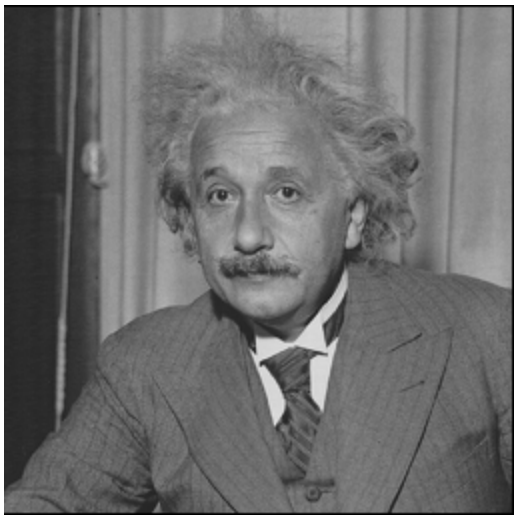


Nearest-Neighbor Einstein (m = 1.3)

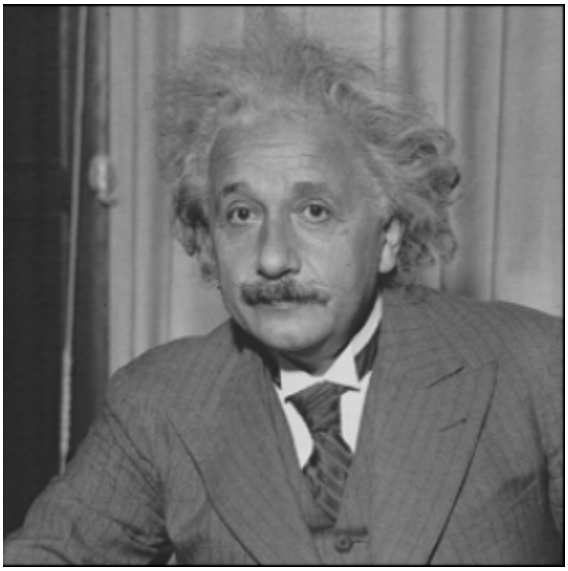


Whereas the edges of Einstein's suit jacket still look reasonably genuine transitioning into the curtain, the white shirt that protrudes from the gray surroundings now assumes an extremely artificial look, much like the cameraman's trench coat. Specifically, the striped tie no longer looks properly striped, and it merges unnaturally with the white shirt, which now appears blocky. Here, with a larger magnification, our eyes find more space to detect irregularities, as we must interpolate more data points; in general, as the image size increases, interpolation artifacts appear more blatantly, since our algorithm must fabricate a larger and larger proportion of the image data. With the high contrast of Einstein's shirt and tie, the discontinuities resulting from nearest-neighbor cut-and-paste quickly manifest themselves, since the grayscale fails to vary continuously, instead assuming one value over a larger area before jumping abruptly. As with the cameraman image, bilinear interpolation maintains a better transition between sharply contrasting regions:

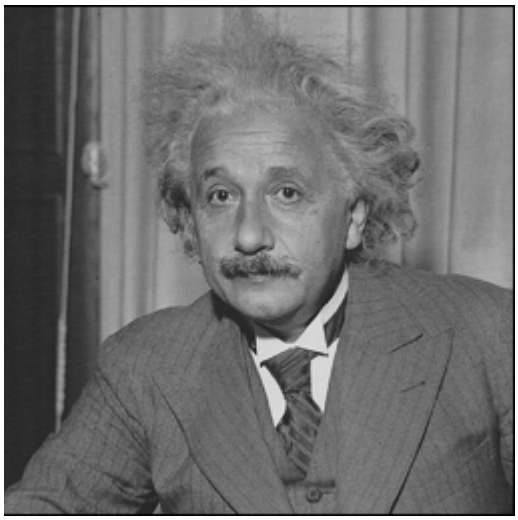
Problem 2.3B - Unmagnified Einstein



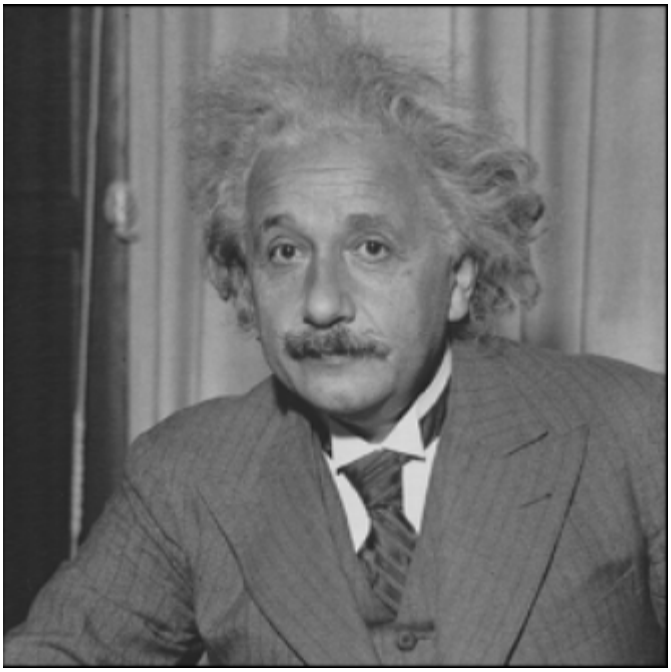
Bilinear Interpolation Einstein (m = 1.1)



Problem 2.3B - Unmagnified Einstein



Bilinear Interpolation Einstein (m = 1.3)



In the larger magnification, Einstein's tie loses some of its fine detail due to image expansion, but no jagged edges taint the subjective quality of either magnification. Thus, we conclude that bilinear interpolation, though computationally more complex than nearest-neighbor interpolation, produces a more visually pleasing image, since boundaries separating two contrasting regions preserve the expected linearity (or curvature) by incorporating all neighboring pixel values into the computation.