1. This problem investigates nearest neighbor and bilinear interpolation. For simplicity, we will focus on estimating the image intensity at a single location. Interpolation is used when transforming an image through resizing, rotating, etc. in which case, the image intensity will need to be estimated at a number of locations.

Consider the diagram below of four pixels

where the dots (●) represent the locations where we know the image intensity and the × represents the location where we would like to estimate the image intensity. By convention, the vertical axis is the x-axis and the horizontal axis is the y-axis.

Suppose the four pixels are at the following locations (indicated by \((x,y)\)) and have the following intensity values (indicated by \(p\))

\[
\begin{align*}
(x_1, y_1) &= (4,10) & p_1 &= 100 \\
(x_2, y_2) &= (4,11) & p_2 &= 107 \\
(x_3, y_3) &= (5,10) & p_3 &= 120 \\
(x_4, y_4) &= (5,11) & p_4 &= 130
\end{align*}
\]

and that we would like to estimate the image intensity at a fifth location

\[
(x_5, y_5) = (4.3,10.4)
\]

That is, we want to estimate \(p_5\).

a) Provide an estimate for \(p_5\) using nearest neighbor interpolation.
b) Provide an estimate for $p_5$ using bilinear interpolation. Round your value to the nearest integer. You can use either of the two methods discussed in lecture. (You might want to use both methods to check your answer.)

2. Problem 2.11 (page 101) in the 3rd edition of the text or problem 2.16 (page 124) in the 4th edition of the text.


4. Problem 2.18 (page 102) in the 3rd edition of the text or problem 2.24 (page 125) in the 4th edition of the text.

5. Problem 2.19 (page 102) in the 3rd edition of the text or problem 2.26 (page 125) in the 4th edition of the text.

6. This problem will help with your project on image resizing.

Geometric transformations of an image, such as resizing, produce images whose pixels typically don’t align with pixels in the original image. This raises the question of what values to assign to the pixels in the transformed image. This can be accomplished using interpolation. Most interpolation approaches estimate the pixel values in the transformed image using the values of the “closest” pixels in the original image.

Note that rethinking the transformation helps with solving this problem. All we really want are the pixel values in the transformed image. So, rather than computing where the pixels in the original image are mapped to, such as in the diagram below,
we instead compute where the pixels in the transformed image are mapped from, as in the diagram below,

![Image of pixel mapping](image)

Thus, we actually perform the inverse mapping.

Once we have computed where the pixel is mapped from, we can use the values of surrounding pixels in the original image (whose values we know) to estimate the value of the pixel in the transformed image. Nearest neighbor interpolation uses the value of the closest pixel. Bilinear interpolation uses the values of the four closest pixels. In this problem, you will explore how to determine the locations of these pixels (not the values).

To simplify the problem, we will work in 1D. Instead of determining the closest pixels in an image, we will only consider the closest pixels on a line. The extension to 2D should be straightforward. We will also only consider resizing as the geometric transformation.

We will model a 1D M-pixel “image” as follows

![1D pixel locations and boundaries](image)

Now, the inverse mapping of a pixel in a transformed image of size $M'$ to an original image of size $M$ can be viewed as follows
a) Find the linear mapping from a location \( m' \) in the transformed image to the location \( x \) in the original image. That is, derive a function that given \( m' \), computes \( x \). The transformed image measures \( M' \) pixels and the original image measures \( M \) pixels. Note that \( M' \) can be greater than or less than \( M \).

Your mapping should have the following form

\[
x = t(m') = \frac{A}{B} (m' - C) + D
\]

where the constants \( A, B, C, D \) are determined from the following constraints:

- The left boundary of the transformed image should map to the left boundary of the original image.
- The right boundary of the transformed image should map to the right boundary of the original image.
- Locations in between should be mapped linearly (proportionally).

(One way to check if you have the correct equation is to see whether it satisfies the first two constraints above.)

Note that the linear equation above really only has two constants and could be written as

\[
x = t(m') = Am' + B.
\]

I provided the four-constant version above to help you think about how to incorporate the constraints (that form helps me, at least). You can use either form in your answer.
Note that \( x \) typically won’t be integer valued (that is, it won’t fall on a pixel location in the original image) even if \( m' \) is integer valued.

b) Derive the logic and computation to determine the closest pixel \( m \) to a location \( x \) in the original image. That is, given \( x \), determine \( m \) where \( m \in [1,\ldots,M] \). You can assume

\[
0.5 \leq x \leq M + 0.5.
\]

c) Derive the logic and computation to determine the two closest pixels \( m_1 \) and \( m_2 \) to a location \( x \) in the original image. Some special cases you might need to consider:

- \( x \) is integer valued
- \( x \) is less than 1.0
- \( x \) is greater than \( M \)
- \( x \) is equal to 1.0
- \( x \) is equal to \( M \)

d) Now, think about the 2D case. Both dimensions will need to be mapped from the transformed image to the original image. Nearest neighbor interpolation still only requires the closest pixel. Bilinear interpolation now requires the four closest pixels.

Suppose mapping a pixel in the transformed image along the \( m \) dimension results in the locations \( m_1 \) and \( m_2 \) being the \( m \)-coordinates of the closest points, and that mapping a pixel in the transformed image along the \( n \) dimension results in the locations \( n_1 \) and \( n_2 \) being the \( n \)-coordinates of the closest points. List the coordinates of the four closest points in 2D. (This should be fairly straightforward but I want to get you thinking about the problem in 2D for your project).