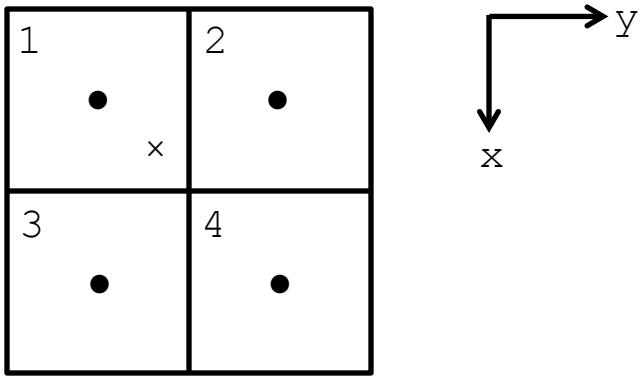
Homework #2
SOLUTION

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)

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

   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.

SOLUTION:

Nearest neighbor interpolation assumes the digital image is a sampled version of a continuous image that has constant intensity from a pixel center to the boundaries of the adjacent pixels. Therefore, to estimate the value at a particular location, we first determine which pixel boundary it falls in and then assign the value from that pixel.

To do this, we simply use the value of the nearest pixel (and thus the name of the method). In our case, the nearest pixel to location × is pixel 1 which has a value of 100. So,

$$p_5 = 100.$$

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.)

SOLUTION:

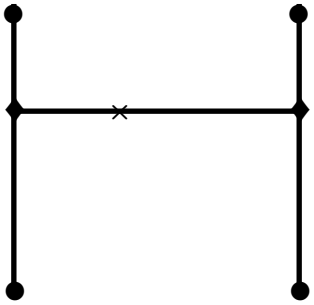
Nearest neighbor interpolation assumes the digital image is a sampled version of a continuous image with a bilinear surface. That is, the surface varies linearly along each of the x and y axes, or, equivalently, has the parametric form:

$$f(x, y) = ax + by + cxy + d \quad .$$

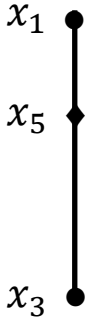
Method 1:

The first method presented in lecture performs linear interpolation in stages, first along one dimension and then the other. The order does not matter.

We will first interpolate along the x-axis to estimate the values at the locations indicated by the diamonds (♦) in the figure below and then interpolate along the y-axis to estimate the value at the location ×, which is what we want.

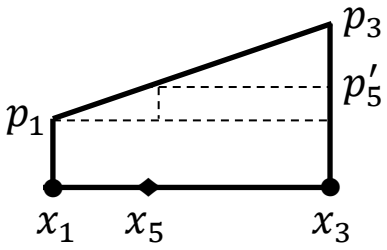


We have the following linear situation for pixels 1 and 3



where location x_1 has value p_1, location x_3 has value p_3, and we want to estimate the value at location x_5 which we will denote as p_5' .

Looking at this "sideways", where height is the image intensity, we have



From similar triangles, we have

$$\frac{p_5' - p_1}{x_5 - x_1} = \frac{p_3 - p_1}{x_3 - x_1}$$

or,

$$p_5' = (p_3 - p_1)\frac{(x_5 - x_1)}{(x_3 - x_1)} + p_1 .$$

Substituting for the known values, $x_1 = 4$, $x_3 = 5$, $x_5 = 4.3$, $p_1 = 100$, and $p_3 = 120$, we get

$$p_5' = 106 .$$

Repeating for pixels 2 and 4

where location x_2 has value p_2, location x_4 has value p_4, and we want to again estimate the value at location x_5 which we will denote as p_5'' .

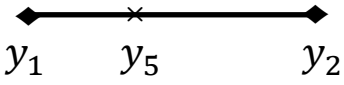
Using similar triangles, we end up with

$$p_5'' = (p_4 - p_2)\frac{(x_5 - x_2)}{(x_4 - x_2)} + p_{21} .$$

Substituting for the known values, $x_2 = 4$, $x_4 = 5$, $x_5 = 4.3$, $p_2 = 107$, and $p_4 = 130$, we get

$$p_5'' = 113.9 .$$

We now have the following linear situation for the two locations that we just estimated the value at (represented by diamonds):



Using similar triangles, we end up with

$$p_5 = (p_5'' - p_5')\frac{(y_5 - y_1)}{(y_2 - y_1)} + p_5' .$$

Substituting for the known values, $y_1(= y_3) = 10$, $y_2(= y_4) = 11$, $y_5 = 10.4$, $p_5' = 106$, and $p_5'' = 113.9$, we get

$$p_5 = 109.16$$

or, rounding to the nearest integer

$$p_5 = 109\,.$$

Method 2:

Here, we first determine the parameters *a*, *b*, *c*, and *d* in the bilinear equation

$$f(x, y) = ax + by + cxy + d \quad .$$

We then can simply use this equation to estimate the value at any location.

Assuming the surface specified by the bilinear equation goes through the four locations at which we know the intensity values, we have:

$$\begin{aligned} p_1 &= ax_1 + by_1 + cx_1y_1 + d \\ p_2 &= ax_2 + by_2 + cx_2y_2 + d \\ p_3 &= ax_3 + by_3 + cx_3y_3 + d \\ p_4 &= ax_4 + by_4 + cx_4y_4 + d \end{aligned}$$

All the values in these equations are known except for the parameters *a*, *b*, *c*, and *d*. This is a system of four equations and four unknowns.

A simple way to solve for the unknowns is to write this system in matrix multiplication form:

$$\begin{bmatrix} x_1 & y_1 & x_1y_1 & 1 \\ x_2 & y_2 & x_2y_2 & 1 \\ x_3 & y_3 & x_3y_3 & 1 \\ x_4 & y_4 & x_4y_4 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

or

$$Ag = h\,.$$

We can solve for g

$$g = A^{-1}h\,.$$

In our case,

$$A = \begin{bmatrix} 4 & 10 & 40 & 1 \\ 4 & 11 & 44 & 1 \\ 5 & 10 & 50 & 1 \\ 5 & 11 & 55 & 1 \end{bmatrix}$$

so,

$$A^{-1} = \begin{bmatrix} -11 & 10 & 11 & -10 \\ -5 & 5 & 5 & -4 \\ 1 & -1 & -1 & 1 \\ 55 & -50 & -44 & 40 \end{bmatrix} .$$

(You can compute this matrix inverse in Matlab.)

And,

$$h = \begin{bmatrix} 100 \\ 107 \\ 120 \\ 130 \end{bmatrix}$$

so,

$$g = \begin{bmatrix} -10 \\ -5 \\ 3 \\ 70 \end{bmatrix}$$

or, $a = -10$, $b = -5$, $c = 3$, and $d = 70$.

We now know the bilinear equation

$$f(x, y) = -10x + (-5)y + 3xy + 70$$

and can estimate the intensity at $(x_5, y_5) = (4.3,10.4)$ as

$$p_5 = 109.16$$

or, rounding to the nearest integer

$$p_5 = 109 .$$

2. Problem 2.16 in the text.

SOLUTION:

Let p and q be as shown in Fig. P2.11. Then, (a) S_1 and S_2 are not 4-connected because q is not in the set $N_4(p)$; (b) S_1 and S_2 are 8-connected because q is in the set $N_8(p)$; (c) S_1 and S_2 are m-connected because (i) q is in $N_D(p)$, and (ii) the set $N_4(p) \cap N_4(q)$ is empty.

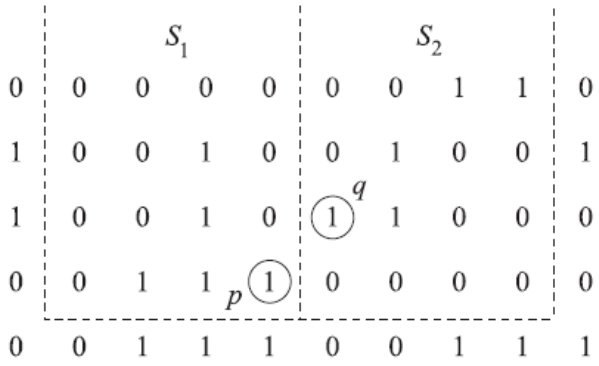


Figure P2.11

3. Problem 2.20 in the text.

SOLUTION:

(a) When $V = \{0, 1\}$, 4-path does not exist between p and q because it is impossible to get from p to q by traveling along points that are both 4-adjacent and also have values from V. Figure P2.15(a) shows this condition; it is not possible to get to q. The shortest 8-path is shown in Fig. P2.15(b); its length is 4. The length of the shortest m- path (shown dashed) is 5. Both of these shortest paths are unique in this case.

(b) One possibility for the shortest 4-path when $V = \{1, 2\}$ is shown in Fig. P2.15(c); its length is 6. It is easily verified that another 4-path of the same length exists between p and q. One possibility for the shortest 8-path (it is not unique) is shown in Fig. P2.15(d); its length is 4. The length of a shortest m-path (shown dashed) is 6. This path is not unique.

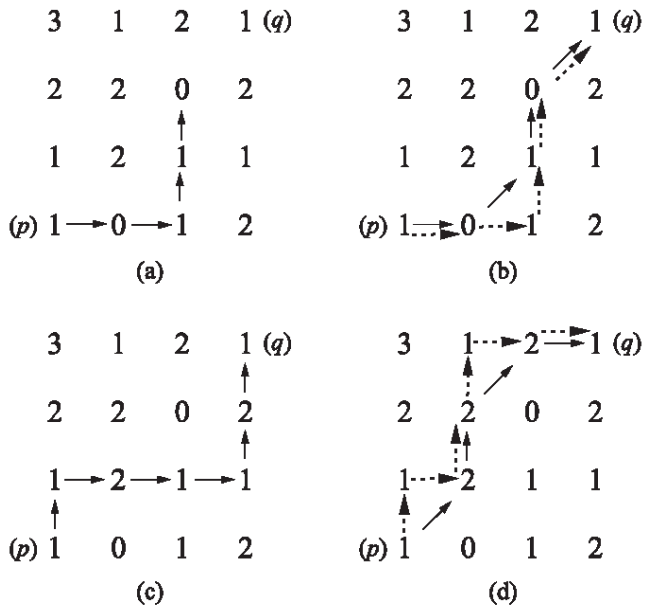


Figure P.2.15

4. Problem 2.24 in the text.

SOLUTION:

With reference to Eq. (2.6-1), let H denote the sum operator, let S_1 and S_2 denote two different small subimage areas of the same size, and let $S_1 + S_2$ denote the corresponding pixel-by-pixel sum of the elements in S_1 and S_2, as explained in Section 2.6.1. Note that the size of the neighborhood (i.e., number of pixels) is not changed by this pixel-by-pixel sum. The operator H computes the sum of pixel values in a given neighborhood. Then, $H(aS_1 + bS_2)$ means: (1) multiply the pixels in each of the subimage areas by the constants shown, (2) add the pixel-by-pixel values from aS_1 and bS_2 (which produces a single subimage area), and (3) compute the sum of the values of all the pixels in that single subimage area. Let ap_1 and bp_2 denote two arbitrary (but *corresponding*) pixels from $aS_1 + bS_2$. Then we can write

$$\begin{aligned} H(aS_1 + bS_2) &= \sum_{p_1 \in S_1 \text{ and } p_2 \in S_2} ap_1 + bp_2 \\ &= \sum_{p_1 \in S_1} ap_1 + \sum_{p_2 \in S_2} bp_2 \\ &= a \sum_{p_1 \in S_1} p_1 + b \sum_{p_2 \in S_2} p_2 \\ &= aH(S_1) + bH(S_2) \end{aligned}$$

which, according to Eq. (2.6-1), indicates that H is a linear operator.

(Note: Eq. (2.6-1) above references the 3rd edition of the text. See Eq. (2-22) in the 4th edition.)

5. Problem 2.26 in the text.

SOLUTION:

The median, ζ, of a set of numbers is such that half the values in the set are below ζ and the other half are above it. A simple example will suffice to show that Eq. (2.6-1) is violated by the median operator. Let $S_1 = \{1, -2, 3\}$, $S_2 = \{4, 5, 6\}$, and $a = b = 1$. In this case H is the median operator. We then have $H(S_1 + S_2) = \text{median}\{5, 3, 9\} = 5$, where it is understood that $S_1 + S_2$ is the array sum of S_1 and S_2. Next, we compute $H(S_1) = \text{median}\{1, -2, 3\} = 1$ and $H(S_2) = \text{median}\{4, 5, 6\} = 5$. Then, because $H(aS_1 + bS_2) \neq aH(S_1) + bH(S_2)$, it follows that Eq. (2.6-1) is violated and the median is a nonlinear operator.

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

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).

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.

SOLUTION:

$$x = t(m') = \frac{M}{M'}(m' - 0.5) + 0.5$$

so that

$$t(0.5) = 0.5$$

and

$$t(M' + 0.5) = M + 0.5.$$

$t(g)$ can also be written as

$$x = t(m') = \left(\frac{M}{M'}\right)m' + \left(0.5 - \frac{M}{M'}(0.5)\right)$$

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,…,M]. You can assume

$$0.5 \le x \le M + 0.5.$$

SOLUTION:

We can simply set $m = \text{round}(x)$. A special case is when $x = M + 0.5$ where we need to set $x = M$.

c) Derive the logic and computation to determine the two closest pixels m1 and m2 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

SOLUTION:

```
If x is integer valued
    m1 = x
    m2 = x
Else
    If x < 1
            m1 = 1
            m2 = 2
    Else if x > m
            m1 = M-1
            m2 = M
    Else
            m1 = floor(x)          // largest integer smaller than x
            m2 = ceiling(x)        // smallest integer larger than x
    End
End
```

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 m1 and m2 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 n1 and n2 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).

SOLUTION:

The four closest pixels are simply

(m1, n1)
(m1, n2)
(m2, n1)
(m2, n2)