Jordan Yono

Homework 1 (Chapter 2)

ELEE-5920

02-03-2021

US Edition: 2.6 = Global Edition 2.4

A. Yes, you can solve the imaging aspects of this problem with a single sensor and camera. In section 2.3, it says "the wavelength of an electromagnetic wave required to 'see' an object must be of the same size as, or smaller than the object". Since the smallest discernible detail on the specimen captured needs to be .001 microns or smaller, this means we need a camera and sensor towards the higher energy end of X-rays, which range from 0.03 and 3 nanometers.
B. N/A

US Edition: 2.8 = Global Edition 2.6

By looking at Figure 2.3, we are given that the height of the tree(15m)/100m(distance away) = h(height of image)/17mm(distance from lens), so the height is 2.55mm. Similarly, for our problem, we have an image height of 7mm(image height)/ 35mm(distance from lens) = 500mm(.05m distance away scaled to mm) / h. Thus, we have 7/35 = 500/h, so h = 100mm. We have 1024 sensing elements, which are targeting a line element with a size of 100mm. Since we are dealing with pairs, each pair of line elements will be 200mm. **We can say that this camera will be able to resolve 1024/200 = 5 line pairs per mm.** *This one took me quite a while because I was stubborn and tried to solve without modeling figure 2.3 to the exact question. After I modeled, the steps became obvious*

US Edition: 2.12 = Global Edition 2.10

We are given 1125 lines horizontally with a width-height aspect ratio of 16/9. If we let x be our vertical line count, then x/1125 = 16/9. Therefore x = 1125 * 16 / 9 = 2000. We have a 2000x1125 resolution image. Since every other line is "painted" on, we have a the total number of pixels we can extract data from is 2000*1125/2 = 1,125,000, we can call this n. Since there are 24 bits in a pixel, we have 2000*1125/2 * 24 bits per frame. Therefore,

*bits = 24 * 2000 * 1125 * 60 (each frame is 1/60th of a second) * 60 (60 seconds in a minute) * 60(minutes in an hour) * 2(total number of hours) =* ***1.1664E+13***.

US Edition: 2.16 = Global Edition 2.14

A. No, No member of S1 is adjacent to the left, right, above, or below any member of S2.
B. Yes
C. Yes

US Edition: 20 = Global Edition 2.18

A. S{0,1} p = (0,3) , q = (3,0)

a. **4-path:** No, because q only has 2 of its 4-path neighbors available and both of those values are 2, thus it is inaccessible in the case where *S{0,1}* and and q's 4-path neighbors are *[(qx - 1, qy - 0) = 2, (qx - 0, qy + 1) = 2].*

3	1	2	(q)1
2	2	**0 (Stranded)**	2
1	2	**1 ^**	1
(p) 1 ->	**0 ->**	**1 ^**	2

b. **8-path:** Yes, you can take the path

(px , py) - > (px+1, py) -> (px + 2, py -1) ->(px + 2, py - 2) -> (px + 3, py - 3) =

(3, 0) = (qx, qy).

This is 4 steps.

3	1	2	**(q)1**
2	2	**0 -> + ^**	2
1	2	**1^**	1
(p) 1 ->	**0 -> + ^**	0	2

c. **M-path:** Yes, you can take the path

(px , py) - > (px+1, py) -> (px + 2, py)-> (px + 2, py -1) ->(px + 2, py - 2) -> (px + 3, py - 3) = (3, 0) = (qx, qy).

This is 5 steps. You can't go diagonally from (px+1, py) -> (px + 2, py -1) like in 8-path. Remember that m-path is a modified version of 8-path, where if a 4 path option exists, you must take it in order to eliminate ambiguity.

3	1	2	**(q)1**
2	2	**0 -> + ^**	2
1	2	**1^**	1
(p) 1 ->	**0 ->**	**0^**	2

B. *S{1,2} p = (0,3) , q = (3,0)*

a. **4-path**: Yes, you can take the path

(px , py) - > (px , py -1) - > (px , py-2) - > (px + 1 , py - 2) - > (px + 1 , py - 3) - > (px + 2 , py - 3) - > (px + 3 , py - 3) = (qx, qy).

This is 6 steps.

3	**1 ->**	**2 ->**	**(q)1**
2 ->	**2 -> + ^**	0 -> + ^	2
1 ^	**2 -> + ^**	1^	1
(p) 1 -> + ^	0 -> + ^	0	2

b. **8-path**: Yes, you can take the path

(px , py) - > (px + 1 , py - 1) -> (px + 1 , py -2) - > (px + 2 , py - 3) -> (px + 3 , py - 3) = (qx, qy).

This is 4 steps.

3	1 ->	**2 ->**	**(q)1**
2 ->	**2 ^**	0 -> + ^	2
1 ^	2	1^	1
(p) 1 ^	0 -> + ^	0	2

c. **M-path:** *(px , py) - > (px , py -1) - > (px , py-2) - > (px + 1 , py - 2) - > (px + 1 , py - 3) - > (px + 2 , py - 3) - > (px + 3 , py - 3) = (qx, qy).*

This is 6 steps.

3	**1 ->**	**2 ->**	**(q)1**
2 ->	**2 -> + ^**	0 -> + ^	2
1 ^	**2 -> + ^**	1^	1
(p) 1 -> + ^	0 -> + ^	0	2

US Edition: 2.25 = Global Edition 2.23

As found in Section 2.6, eq 2-23, we can test if an operator is linear by testing that the sum of two inputs is the same as performing the operation individually on the two inputs and summing the results. It must have additivity and homogeneity. We can use $H[af(x,y) + bg(x,y)] = aH[f(x,y)] + bH[g(x,y)]$ as a base test for this specific problem where a,b are arbitrary constants and H is our generic operator. ***I struggled with A and B quite a bit. I read what's in the book, but it's pretty short and examples are a little limited. I think I understand the general concept, but could you point me towards a better resource for understanding how to prove it? I gave it a good effort, but I don’t feel great about any of them***

A. **Summation is a linear operation**

Given that we have the equation, $s(x,y) = f(x,y) + g(x,y)$, we use two arbitrary constants a, b, and a generic operator H = +, we can show that summation is linear in the formula below because we can see that

$H[af(x,y) + bg(x,y)] = aH[f(x,y)] + bH[g(x,y)]$

$(+)[af(x,y) + bg(x,y)] = a(+)[f(x,y)] + b(+)[g(x,y)]$

$a(+)[f(x,y)] + b(+)[g(x,y)] = a(+)[f(x,y)] + b(+)[g(x,y)]$

As stated above, since the sum of two inputs is the same as performing the operation individually on the two inputs and summing the results, we can say this operation is linear.

B. Subtraction is a linear operation

Incredibly similar to addition, we can keep all things the same, except our generic operator H = - to reflect subtraction. Given that $d(x,y) = f(x,y) - g(x,y)$, we have

$H[af(x,y) + bg(x,y)] = aH[f(x,y)] + bH[g(x,y)]$

$(-)[af(x,y) + b[g(x,y)] = a(-)[f(x,y)] + b(-)[g(x,y)]$

$a(-)[f(x,y)] + b(-)[g(x,y)] = a(-)[f(x,y)] + b(-)[g(x,y)]$

C. Multiplication is a nonlinear operation

The simplest way to prove that multiplication is nonlinear is to find an example of it failing a linear test. Consider that we have one 2x2 matrix *f(x,y) = [0 1; 2 3]* and another one that is *g(x,y) = [6 5; 4 7]*. If we set a=1,b=-1 and evaluate our test, we can see that the operator is nonlinear.

On the left side we have

*([0 1; 2 3]) + [4 5; 6 7]) * ab = [0 5; 8 10] * (1)(-1)*

= [0 -5; -8 -10]

On the right side we have

(a**[0 1; 2 3])+(b*[4 5; 6 7]) =* (1**[0 1; 2 3])+(-1*[4 5; 6 7])*

= [0 1; 2 3] + (-4 -5; -6 -7)

=[-4 -4; -4 -4]

Since the left and right side are the same, we can assume this to be nonlinear.

D. Division is a nonlinear operation

Similar to multiplication, the simplest way to prove that multiplication is nonlinear is to find an example of it failing a linear test. Consider that we have one 2x2 matrix *f(x,y) = [0 1; 2 3]* and another one that is *g(x,y) = [6 5; 4 7]*. If we set a=1,b=-1 and evaluate our test, we can see that the operator is nonlinear.

On the left side we have

([0 1; 2 3]) + [4 5; 6 7]) / ab = [0 5; 8 10] / (1)(-1)

= [0 -5; -8 -10]

On the right side we have

([0 1; 2 3]/a)+([4 5; 6 7]/b) = ([0 1; 2 3] / 1)+(1/[4 5; 6 7] / -1)

= [0 1; 2 3] + (-4 -5; -6 -7)

=[-4 -4; -4 -4]

As you can see, the results of the operations are not equal, and similarly to multiplication, are not linear.

US Edition: 2.29 = Global Edition 2.27

A. The limiting effect of repeatedly subtracting image two from image 1 is that [0-255] is a bounded range and when there is subtraction that would exceed the lower bound, the output will stop at the lower bound. This can cause an obvious distortion because the subtraction that exceeds the lower bound will not maintain idempotency. The same operation with the same input value can have different effects depends on the current state, ie. if the operation exceeds the lower bound or not.

B. Reversing the images would have the same pitfall described above, assuming the repeated subtraction continues to exceed the lower bound.