CIS*4720 Image Processing and Vision

Assignment 3 Part 1 & 2

Maneesh Wijewardhana (1125828)

I have read and understood the Academic Misconduct section in the course outline. I assert this work is my own.

1)

- Since cosine varies between -1 and 1, the maximum value of the cosine term is 1 and the minimum value is -1.
- $\therefore$ the maximum and minimum values of f_a are:
  - $max(f_a) = 255[\frac{1}{2} + \frac{1}{2}(1)] = 255$
  - $min(f_a) = 255[\frac{1}{2} + \frac{1}{2}(-1)] = 0$
  - $\therefore$ the range of f_a is $[0, 255]$
- To show that the edge points lie on a straight line segment S, we need to show that the function f_a is **linear** along the edges of the rectangle $[0, M] \times [0, N]$
- Lets start at the bottom edge of the rectangle, which is the line segment from $(0, 0)$ to $(M, 0)$
- Along this edge, $y = 0$, so the function $f_a(x, 0)$ simplifies to:
  - $f_a(x, 0) = 255[\frac{1}{2} + \frac{1}{2}cos(Mx)$
- Since $cos(Mx)$ is periodic by $2\pi/M$, it has a max value of 1 when $Mx = 2n\pi$ where n is an integer and a min value of -1 when $Mx = (2n + 1)\pi$
- $\therefore f_a(x, 0)$ takes these two values for max and min
- This means that along the bottom edge of the rectangle, the function $f_a(x, 0)$ is linear with slope $(255 - 0)/M = 255/M$
- Similarly, we can show that along the other three edges of the rectangle, the function $f_a(x, y)$ is linear with slopes $255/M, -255/M, 255/N$ respectively
- $\therefore$ the edge points of the rectangle lie on a straight line segment S, which is the line passing through the points $(0, 0, 0), (M, 0, 255/M), (M, N, 255), (0, N, 255/N)$

1a)

- We need to first find the gradient of f_a which is given by $\triangledown f_a(x, y) = [\frac{\partial f_a}{\partial x}(x, y), \frac{\partial f_a}{\partial y}(x, y)]$
- Using the partial derivatives of f_a, we have:
  - $\triangledown f_a(x, y) = [-\frac{255}{2}\frac{M\pi}{M^2+N^2}sin(\frac{Mx+Ny}{M^2+N^2}\pi), -\frac{255}{2}\frac{N\pi}{M^2+N^2}sin(\frac{Mx+Ny}{M^2+N^2}\pi)]$
- Now we can calculate the Euclidean norm of the gradient as follows:
  - $||\triangledown f_a(x, y)|| = \sqrt{((\frac{\partial f_a}{\partial x}(x, y))^2 + (\frac{\partial f_a}{\partial y}(x, y))^2)}$
- Substituting the given partial derivatives, we get:
  - $||\triangledown f_a(x, y)|| = \sqrt{(-\frac{255}{2}\frac{M\pi}{M^2+N^2}sin(\frac{Mx+Ny}{M^2+N^2}\pi))^2 + (-\frac{255}{2}\frac{N\pi}{M^2+N^2}sin(\frac{Mx+Ny}{M^2+N^2}\pi))^2}$
  - Simplifying gets us:
    - $||\triangledown f_a(x, y)|| = (\frac{255\pi}{2} * \sqrt{\frac{M^2+N^2}{M^4+2M^2N^2+N^4}})$
- To find the gradient direction at any edge point, we need to calculate the angle between the gradient vector and the positive x-axis, which is given by:
  - $\theta = arctan(\frac{\partial f_a}{\partial y}(x, y), \frac{\partial f_a}{\partial x}(x, y))$
- Substituting the given partial derivatives, we get:
  - $\theta = arctan(-\frac{255}{2}\frac{M\pi}{M^2+N^2}sin(\frac{Mx+Ny}{M^2+N^2}\pi), -\frac{255}{2}\frac{N\pi}{M^2+N^2}sin(\frac{Mx+Ny}{M^2+N^2}\pi))$
- By simplifying, we get:
  - $\theta = arctan(N, M)$

1b)

- To use the Laplacian approach, we need to find the Laplacian of the function f_a which is defined by the sum of the second partial derivatives of f_a with respect to each variable:
  - $\nabla f_a(x, y) = \frac{\partial^2 f_a}{\partial x^2} + \frac{\partial^2 f_a}{\partial y^2}$
- Using the provided formulas, we can calculate the Laplacian as follows:
- $\nabla f_a(x, y) = -\frac{255M^2\pi^2}{2(M^2+N^2)^2} cos(\frac{Mx+Ny}{M^2+N^2}\pi) - \frac{255N^2\pi^2}{2(M^2+N^2)^2} cos(\frac{Mx+Ny}{M^2+N^2}\pi)$
- Simplifying this expression, we can factor and get:
  - $\nabla f_a(x, y) = -\frac{255\pi^2}{2(M^2+N^2)} cos(\frac{Mx+Ny}{M^2+N^2}\pi)(M^2 + N^2)$
- $\therefore$ the Laplacian of f_a is a multiple of $cos(\frac{Mx+Ny}{M^2+N^2}\pi)$, which means that the Laplacian is positive when $cos(\frac{Mx+Ny}{M^2+N^2}\pi)$ is negative and negative when it is positive

2a)

```
boolean are4Connected(element p, element q, set A):
    # if p or q is not in A, cannot be 4-connected
    if not belongsTo(p, A) or not belongsTo(q, A):
        return False

    # has to be 4-connected in this case
    if p == q:
        return True

    # find all 4 components of A
    fourComponents = 4componentNumber(number4Components(A), A)

    # check if p and q are in the same 4-connected component
    for component in fourComponents:
        if belongsTo(p, component) and belongsTo(q, component):
            return True
    return False
```

2b)

```
boolean is4Connected(set A):
    # if empty or 1 element, it is 4 connected
    if A == emptySet() or numberElements(A) == 1:
        return True

    # then check number of 4 components to determine connectedness
    numComp = number4Components(A)
    if numComp == 1:
        return True
    else:
        return False
```

2c)

```
boolean are4Adjacent(element p, element q, set A):
    neighbors = 4neighbours(p)
    if belongsTo(q, neighbors) and belongsTo(q, A):
        return True
    else:
        return False
```

2d)

```
boolean are4Adjacent(set A, set B):
    # need to compare all elements in A with all elements in B
    for i = 1 to numberElements(A):
        # check if current is in a connected component
        # if so, check if it belongs to component in B
        for j = 1 to number4Components(A):
            if belongsTo(elementNumber(i, A), 4componentNumber(j, A)):
                for k = 1 to numberElements(B):
                    for l = 1 to number4Components(B):
                        if belongsTo(elementNumber(k, B), 4componentNumber(l, B)):
                            # now check if they are neighbors
                            if belongsTo(elementNumber(k, B), 4neighbors(elementNumber(i, A))):
                                return True
    return False
```

3a)

```
integer numberHoles8(set A):
    B = complement(A)
    numB = number8Components(B)
    numA = number8Components(A)
    numHoles = numB - numA
    return numHoles
```

3b)

```
void fillHoles8(set A):
    # get all holes and the universe
    B = complement(A)

    # then get number of 8components of this set
    num8Components = number8Components(B)

    # now loop and if numElements does not return -1 (infinite), then its a hole
    for i = 1 to num8Components:
        8compNum = 8componentNumber(i, B)
        if numberElements(8compNum) != -1:
            union(A, 8compNum)
```

3c)

```
set boundary8(set A):
    boundary = emptySet()
    neighbors = emptySet()
    allNeighbors = emptySet()

    for element in A:
        neighbors = 8neighbours(element)
        for neighbor in neighbors:
            if not belongsTo(neighbor, A):
                addToSet(neighbor, boundary)
        addToSet(neighbors, allNeighbors)

    # subract set of elements in A from set of neighbors to obtain set of exterior neighbors
    subtractFromSet(allNeighbors, A)

    # subtract set of elements in A from the set of boundary elements to obtain boundary set
    subtractFromSet(boundary, A)
    return boundary
```