Lecture 3

Image Fundamentals II

Outline

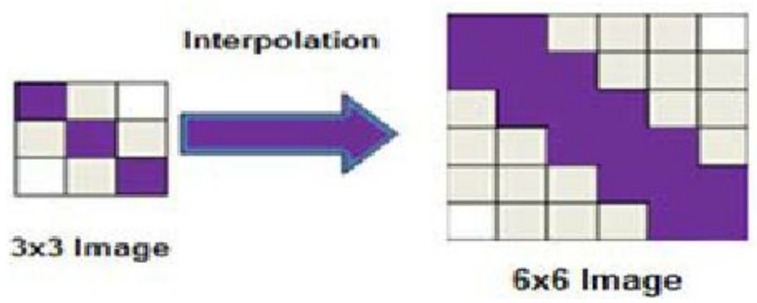
Image Interpolation

Pixel Relationships

Interpolation is used in tasks such as zooming, shrinking, rotating, and geometrically correcting digital images.

Interpolation is the process of using known data to estimate values at unknown locations.

- nearest neighbor interpolation
- bilinear interpolation
- bicubic interpolation



Nearest Neighbor Interpolation

Suppose you're scaling an image up or down.

For each pixel in the new (output) image, you **map it back** to a coordinate in the original (input) image using the inverse transformation.

You then find the **nearest integer coordinate** (i.e., the nearest pixel) in the original image and **copy its value** directly.

Nearest Neighbor Interpolation

The idea is to (i) find the four neighbors in the original image, (ii) take the intensity value of the most nearest one, and (iii) assign it to the pixel concerned in the new image.

(x,y)->(u,v) via Scaling operation

Inverse scaling: (u*X/U, v*Y/V)

N_1=(floor(u*X/U),floor(v*Y/V))

N_2=(floor(u*X/U),ceil(u*Y/V))

N_3=(ceil(u*X/U),floor(u*Y/V))

N_4=(ceil(u*X/U),ceil(u*Y/V))

NN= nearest neighbor of (u*X/U,v*Y/V)

I_n(u,v)=I_o(NN)

(x,y): pixel location in original image (I_o) of size X*Y
(u,v): pixel location in new image (I_n) of size U*V

Very fast and computationally inexpensive

Easy to implement

Bilinear Interpolation

$I_o(x,y)=ax+by+cxy+d$

Substitute the 4 neighbors' values (pixel locations and intensities) and solve for (a,b,c and d)

Once we have (a,b,c and d), substitute $x=u*X/U$ and $y=v*Y/V$ in $I_o(x,y)$ to get $I_n(u,v)$

New image

It's an alternative representation for the following

$$f(x,y) \approx \frac{1}{(x_2-x_1)(y_2-y_1)} \cdot \left[Q_{11}(x_2-x)(y_2-y) + Q_{21}(x-x_1)(y_2-y) + Q_{12}(x_2-x)(y-y_1) + Q_{22}(x-x_1)(y-y_1) \right]$$

Bicubic Interpolation

$$I_o(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} x^i y^j$$

Same concept as bilinear interpolation, but we now have 16 neighbors.

All the 16 pixels between (floor(u*X/U)-1,floor(v*Y/V)-1) and (ceil(u*X/U)+1,ceil(v*Y/V)+1)

(floor(u*X/U)-1, floor(v*Y/V)-1)	(floor(u*X/U)-1, floor(v*Y/V))	(floor(u*X/U)-1, ceil(v*Y/V))	(floor(u*X/U)-1, ceil(v*Y/V)+1)
(floor(u*X/U), floor(v*Y/V)-1)	(floor(u*X/U), floor(v*Y/V))	(floor(u*X/U), ceil(v*Y/V))	(floor(u*X/U), ceil(v*Y/V)+1)
(ceil(u*X/U), floor(v*Y/V)-1)	(ceil(u*X/U), floor(v*Y/V))	(ceil(u*X/U), ceil(v*Y/V))	(ceil(u*X/U), ceil(v*Y/V)+1)
(ceil(u*X/U)+1, floor(v*Y/V)-1)	(ceil(u*X/U)+1, floor(v*Y/V))	(ceil(u*X/U)+1, ceil(v*Y/V))	(ceil(u*X/U)+1, ceil(v*Y/V)+1)

Neighbors of Pixels

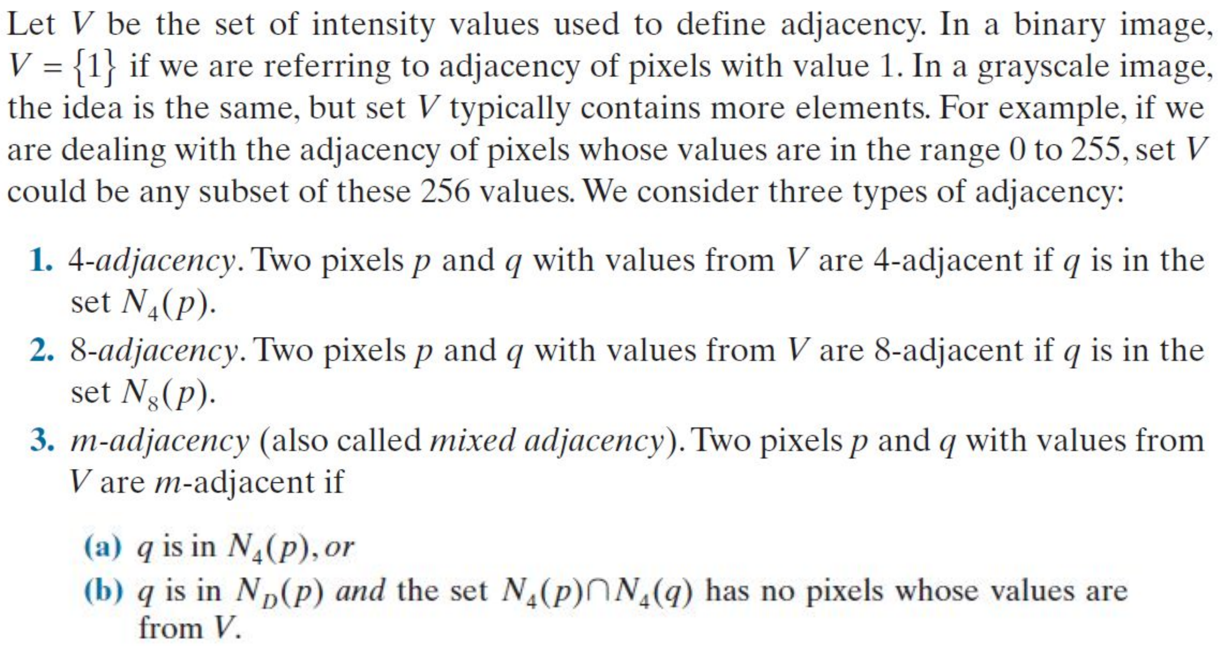
pixel p at coordinates (x, y)

$$N_4(p) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}$$

$$N_D(p) = \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\}$$

$$N_8(p) = N_4(p) \cup N_D(p)$$

Adjacency



Let V be the set of intensity values used to define adjacency. In a binary image, $V = \{1\}$ if we are referring to adjacency of pixels with value 1. In a grayscale image, the idea is the same, but set V typically contains more elements. For example, if we are dealing with the adjacency of pixels whose values are in the range 0 to 255, set V could be any subset of these 256 values. We consider three types of adjacency:

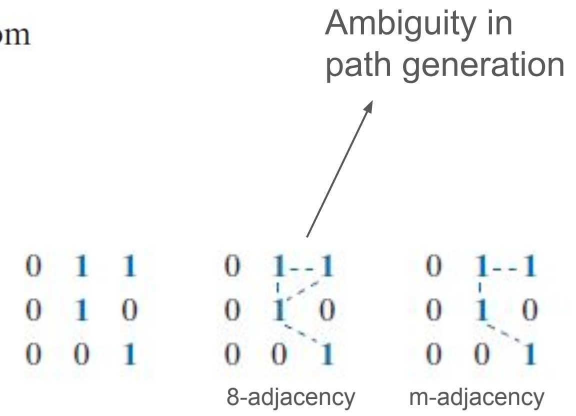
1. 4-*adjacency*. Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$.

2. 8-*adjacency*. Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.

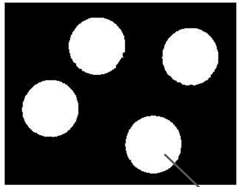
3. *m-adjacency* (also called *mixed adjacency*). Two pixels p and q with values from V are m-adjacent if

  (a) q is in $N_4(p)$, *or*

  (b) q is in $N_D(p)$ *and* the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V.



Connected Component



4 connected components in this image (subset of only 1s)

connected component

A *digital path* (or *curve*) from pixel p with coordinates (x_0, y_0) to pixel q with coordinates (x_n, y_n) is a sequence of distinct pixels with coordinates

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

where points (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$. In this case, n is the *length* of the path. If $(x_0, y_0) = (x_n, y_n)$ the path is a *closed* path. We can define 4-, 8-, or m-paths, depending on the type of adjacency specified.

Let S represent a subset of pixels in an image. Two pixels p and q are said to be *connected in S* if there exists a path between them consisting entirely of pixels in S. For any pixel p in S, the set of pixels that are connected to it in S is called a *connected component* of S.