Distance measures

• Given pixels $p$, $q$, and $z$ at $(x,y)$, $(s,t)$ and $(u,v)$ respectively,

• $D$ is a distance function (or metric) if:
  – $D(p,q) \geq 0$ (if $p=q$),
  – $D(p,q) = D(q,p)$, and
  – $D(p,z) \leq D(p,q) + D(q,z)$.

• The Euclidean distance between $p$ and $q$ is given by:

$$D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$$

• The pixels having distance less than or equal to some value $r$ from $(x,y)$ are the points contained in a disk of radius $r$ centered at $(x,y)$
Distance measures (continued)

• The D₄ distance (also called the *city block distance*) between p and q is given by:
  \[ D₄(p, q) = |x - s| + |y - t| \]

• The pixels having a D₄ distance less than some r from (x,y) form a diamond centered at (x,y)

• Example: pixels where D₄ ≤ 2

```
  2 1 2 1 2 1 2
2 1 0 1 2 1 2
  2 1 2 2 1 2
  2 2 2 2 2 2
```

Note: Pixels with D₄=1 are the 4-neighbors of (x,y)

Distance measures (continued)

• The D₈ distance (also called the *chessboard distance*) between p and q is given by:
  \[ D₈(p, q) = \max(|x - s|, |y - t|) \]

• The pixels having a D₈ distance less than some r from (x,y) form a square centered at (x,y)

• Example: pixels where D₈ ≤ 2

```
  2 2 2 2 2 2 2
2 1 1 1 1 1 2
2 1 0 1 2 1 2
2 1 1 1 1 1 2
2 2 2 2 2 2 2
```

Note: Pixels with D₈=1 are the 8-neighbors of (x,y)
Distance measures and connectivity

- The D₄ distance between two points p and q is the shortest 4-path between the two points
- The D₈ distance between two points p and q is the shortest 8-path between the two points
- D₄ and D₈ may be considered, regardless of whether a connected path exists between them, because the definition of these distances involves only the pixel coordinates
- For m-connectivity, the value of the distance (the length of the path) between two points depends on the values of the pixels along the path

Distance measures and m-connectivity

- Consider the given arrangement of pixels and assume
  - p, p₂ and p₄ = 1
  - p₁ and p₃ can be 0 or 1
- If V={1} and p₁ and p₃ are 0, the m-distance (p, p₄) is 2
- If either p₁ or p₃ are 1, the m-distance (p, p₄) is 3
- If p₁ and p₃ are 1, the m-distance (p, p₄) is 4
**M-connectivity example**

```
  0 1
  0 1
  1 1
  1 1

  m-distance(p_1,p_2)=2
  m-distance(p_1,p_4)=3
  m-distance(p_2,p_4)=4
```

---

**Arithmetic & logic operations**

- Arithmetic & logic operations on images used extensively in most image processing applications
  - May cover the entire image or a subset
- Arithmetic operation between pixels \( p \) and \( q \) are defined as:
  - Addition: \((p+q)\)
    - Used often for image averaging to reduce noise
  - Subtraction: \((p-q)\)
    - Used often for static background removal
  - Multiplication: \((p\times q)\) (or \( pq \)
    - Used to correct gray-level shading
  - Division: \((p\div q)\) (or \( p/q \)
    - As in multiplication
Logic operations

- Arithmetic operation between pixels $p$ and $q$ are defined as:
  - AND: $p \land q$ (also $p \cdot q$)
  - OR: $p \lor q$ (also $p + q$)
  - COMPLEMENT: $\neg q$ (also $q'$)
- Form a *functionally complete* set
- Applicable to binary images
- Basic tools in binary image processing, used for:
  - Masking
  - Feature detection
  - Shape analysis

Examples of logic operations

\[
\begin{array}{ccc}
A & \text{NOT}(A) & (A) \land (B) \\
\begin{array}{c}
\text{white} \\
\text{black}
\end{array} & \begin{array}{c}
\text{black} \\
\text{white}
\end{array} & \begin{array}{c}
\text{black} \\
\text{white}
\end{array}
\end{array}
\]
Examples of logic operations (continued)

\[
\begin{array}{ccc}
A & B & (A) \text{ XOR} \ (B) \\
\text{[Black]} & \text{[Black]} & \text{[Gray]} \\
\end{array}
\]

\[
\begin{array}{ccc}
A & B & [\text{NOT (A)}] \text{ AND} \ (B) \\
\text{[Black]} & \text{[Black]} & \text{[White]} \\
\end{array}
\]

Neighborhood-oriented operations

- Arithmetic and logical operations may take place on a subset of the image
  - Typically neighborhood oriented
- Formulated in the context of *mask* operations (also called *template*, *window* or *filter* operations)
- Basic concept: let the value of a pixel be a function of its (current) gray level and the gray level of its neighbors (in some sense)
Neighborhood-oriented operations (continued)

- Consider the following subset of pixels in an image
- Suppose we want to filter the image by replacing the value at $Z_5$ with the average value of the pixels in a 3x3 region centered around $Z_5$
- Perform an operation of the form:

$$z = \frac{1}{9}(z_1 + z_2 + \ldots + z_9) = \frac{1}{9} \sum_{i=1}^{9} z_i$$

- and assign to $z_5$ the value of $z$

Neighborhood-oriented operations (continued)

- In the more general form, the operation may look like:

$$z = (w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9) = \sum_{i=1}^{9} w_i z_i$$

- This equation is widely used in image processing
- Proper selection of coefficients (weights) allows for operations such as
  - noise reduction
  - region thinning
  - edge detection
The sample image was filtered using a 3x3 basic high pass filter as follows:

\[
\frac{1}{9} \begin{bmatrix}
  -1 & -1 & -1 \\
  -1 & 8 & -1 \\
  -1 & -1 & -1 \\
\end{bmatrix}
\]

Result showing highlighted edges in the image.

The sample image was filtered using a 3x3 basic low pass filter as follows:

\[
\frac{1}{9} \begin{bmatrix}
  1 & 1 & 1 \\
  1 & 1 & 1 \\
  1 & 1 & 1 \\
\end{bmatrix}
\]

Result showing filtered image.
Sample filtered image

- The sample image was filtered using a 5x5 basic low pass filter as follows:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

- Result showing filtered image

Basic matlab instructions

- Sample matlab source files and data files available via website
  - `bmpread.m`: A matlab source file which will load a windows bitmap file into an array (256 color maximum)
    - Format:
      \[
      [A\_NAME, CMAP\_NAME]=bmpread('filename');
      \]
      - Creates an array (with name `A\_NAME`) to hold the pixel data
      - Creates a colormap (with name `CMAP\_NAME`)
  - `bmpwrite.m`: A matlab source file which will save an array to a windows bitmap file (256 color maximum)
    - Format:
      \[
      bmpwrite(A\_NAME, CMAP\_NAME,'filename');
      \]
Basic matlab instructions (continued)

- `colormap(CMAP_NAME);` Use a particular colormap to display an image
- `image(A_NAME);` create and display an image from a given array of data