Arithmetic/Logic Operations

Prof. George Wolberg
Dept. of Computer Science
City College of New York
Objectives

• In this lecture we describe arithmetic and logic operations commonly used in image processing.
• Arithmetic ops:
  - Addition, subtraction, multiplication, division
  - Hybrid: cross-dissolves
• Logic ops:
  - AND, OR, XOR, BIC, …
Arithmetic/Logic Operations

- Arithmetic/Logic operations are performed on a pixel-by-pixel basis between two images.
- Logic NOT operation performs only on a single image.
  - It is equivalent to a negative transformation.
- Logic operations treat pixels as binary numbers:
  - \(158 \& 235 = 10011110 \& 11101011 = 10001010\)
- Use of LUTs requires 16-bit rather than 8-bit indices:
  - Concatenate two 8-bit input pixels to form a 16-bit index into a 64K-entry LUT. Not commonly done.
Addition / Subtraction

Addition:

\[
\text{for}(i=0; \ i<\text{total}; \ i++)
\]
\[
\quad \text{out}[i] = \text{MIN}((\text{int})\text{in1}[i]+\text{in2}[i]), \ 255); 
\]

Subtraction:

\[
\text{for}(i=0; \ i<\text{total}; \ i++)
\]
\[
\quad \text{out}[i] = \text{MAX}((\text{int})\text{in1}[i]-\text{in2}[i]), \ 0); 
\]

Avoid overflow: clip result

Avoid underflow: clip result
Overflow / Underflow

- Default datatype for pixel is unsigned char.
- It is 1 byte that accounts for nonnegative range [0,255].
- Addition of two such quantities may exceed 255 (overflow).
- This will cause wrap-around effect:
  - 254: 11111110
  - 255: 11111111
  - 256: 100000000
  - 257: 100000001
- Notice that low-order byte reverts to 0, 1, … when we exceed 255.
- Clipping is performed to prevent wrap-around.
- Same comments apply to underflow (result < 0).
Implementation Issues

• The values of a subtraction operation may lie between -255 and 255. Addition: [0,510].
• Clipping prevents over/underflow.
• Alternative: scale results in one of two ways:
  1. Add 255 to every pixel and then divide by 2.
     • Values may not cover full [0,255] range
     • Requires short intermediate image
     • Fast and simple to implement
  2. Add negative of min difference (shift min to 0). Then, multiply all pixels by 255/(max difference) to scale range to [0,255] interval.
     • Full utilization of [0,255] range
     • Requires short intermediate image
     • More complex and difficult to implement
Example of Subtraction Operation

**FIGURE 3.28**  
(a) Original fractal image.  
(b) Result of setting the four lower-order bit planes to zero.  
(c) Difference between (a) and (b).  
(d) Histogram-equalized difference image.  
(Original image courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA).
Example: Motion Detection

• Use **frame differencing** to compare successive video frames
• Insight: since camera is stationary, the background will not change much
• The moving foreground will change considerably
• Basic approach: use two successive frames to compute a **difference image**
• This produces a binary image from the threshold of $|I_1-I_2|$
• Unfortunately this two-frame approach suffers from the double-image problem, which will display foreground pixels in both current and adjacent frame
• Solution: use double difference image (or three-frame difference)
**Frame Differencing**

**Figure 3.27** Detecting a moving object by frame differencing. **LEFT COLUMN:** Three image frames from a video sequence. **SECOND COLUMN:** The absolute difference between pairs of frames. **THIRD COLUMN:** Thresholded absolute difference. **RIGHT COLUMN:** Final result using double difference (top), triple difference (middle), and thresholded triple difference (bottom) methods.

Wolberg: Image Processing Course Notes
Pseudocode: Double Differencing

**Algorithm 3.13** Compute the double difference between three consecutive image frames

```plaintext
FRAMEDIFFERENCEDOUBLE \( (I_{t-1}, I_t, I_{t+1}, \tau) \)

Input: successive images \( I_{t-1}, I_t, \) and \( I_{t+1}, \) and threshold \( \tau \)
Output: binary image indicating the moving regions

1. for \( (x, y) \in I_t \) do
2. \( d_1 \leftarrow |I_{t-1}(x, y) - I_t(x, y)| \)
3. \( d_2 \leftarrow |I_{t+1}(x, y) - I_t(x, y)| \)
4. \( I'(x, y) \leftarrow 1 \text{ if } d_1 > \tau \text{ and } d_2 > \tau \text{ else } 0 \)
5. return \( I' \)
```
**Pseudocode: Triple Differencing**

**Algorithm 3.14** Compute the triple difference between three consecutive image frames

\[
\text{FRAMEDIFFERENCETRIPLE} \left( I_{t-1}, I_t, I_{t+1}, \tau \right)
\]

**Input:** successive images \( I_{t-1}, I_t, \) and \( I_{t+1}, \) and threshold \( \tau \)

**Output:** binary image indicating the moving regions

1. \( \text{for} \ (x, y) \in I_t \ \text{do} \)
2. \( d_1 \leftarrow |I_{t-1}(x, y) - I_t(x, y)| \)
3. \( d_2 \leftarrow |I_{t+1}(x, y) - I_t(x, y)| \)
4. \( d_3 \leftarrow |I_{t+1}(x, y) - I_{t-1}(x, y)| \)
5. \( I'(x, y) \leftarrow 1 \text{ if } d_1 + d_2 - d_3 > \tau \ \text{else} \ 0 \)
6. \( \text{return } I' \)
Example: Mask Mode Radiography

- \( h(x,y) \) is the mask, an X-ray image of a region of a patient’s body captured by an intensified TV camera (instead of traditional X-ray film) located opposite an X-ray source.
- \( f(x,y) \) is an X-ray image taken after injection a contrast medium into the patient’s bloodstream.
- Images are captured at TV rates, so the doctor can see how the medium propagates through the various arteries in an animation of \( f(x,y) - h(x,y) \).

**Note:**
- The background is dark because it doesn’t change much in both images.
- The difference area is bright because it has a big change.
Arithmetic Operations: Cross-Dissolve

- Linearly interpolate between two images.
- Used to perform a fade from one image to another.
- Morphing can improve upon the results shown below.

```c
for(i=0; i<total; i++)
    out[i] = in1[i]*f + in2[i]*(1-f);
```

![Cross-Dissolve Diagram](image.png)
Masking

• Used for selecting subimages.
• Also referred to as region of interest (ROI) processing.
• In enhancement, masking is used primarily to isolate an area for processing.
• AND and OR operations are used for masking.
Example of AND/OR Operation