Outline

• What is image blending?
• Basic arithmetic operations:
  • Subtraction/difference
  • Average/weighted average
  • Logic operations
• Image compositing:
  • Foreground mask
  • Alpha channel
  • Blending functions

Subtraction/Difference

• Calculate per-pixel (absolute) difference
  • Highlights differences between two images
  \[ G(x, y) = |F(x, y) - E(x, y)| \]
  \[ G(x, y) = F(x, y) - E(x, y) + c \]

Average/Weighted Average

• Calculate per-pixel weighted average
  • Blend between the two input images
  \[ G(x, y) = \left( F(x, y) + E(x, y) \right) / 2 \]
  \[ G(x, y) = (1 - t)F(x, y) + tE(x, y) \]
**Logic Operations**
- Perform logic operations on pixel-by-pixel basis
- Only need to implement AND, OR, & NOT
- Others can be implemented using these 3 above
- NOT operator performs the same as negative transformation

**Extract ROI using AND/OR**
- Both AND & OR operators can extract region of interest (ROI)
- Which one to use depends on how the mask is defined

**Image Compositing**
- Objective:
  - Combine multiple images as overlapping layers to produce a single output image
- Applications:
  - Carton animation
  - Blue screen matting

**Foreground Mask**
- Use a binary mask to indicate the location & shape of the foreground object
- Widely used in graphics-user interface
  - Irregular shape window

**Compositing with Foreground Mask**
- Algorithm:
  - \( R = \{ M = 0 \} \cap R_f; R_b \)
  - \( G = \{ M = 0 \} \cap G_f; G_b \)
  - \( B = \{ M = 0 \} \cap B_f; B_b \)
- Problem:
  - No partial coverage, resulting aliasing around boundary
  - Cannot represent semi-transparent object

**Alpha Channel**
- Indicates the level of transparency or coverage
  - \( \alpha = 0 \)
    - No coverage
    - Transparent
  - \( \alpha = 1 \)
    - Full coverage
    - Opaque
  - \( 0 < \alpha < 1 \)
    - Partial coverage
    - Semi-transparent
Blending with Alpha Channel

- Referred as “over” operator
- Proposed by Smith & Catnull
- Linearly interpolate between foreground & background colors
- Algorithm:
  \[ R = R_s(1-a_f) + R_f a_f \]
  \[ G = G_s(1-a_f) + G_f a_f \]
  \[ B = B_s(1-a_f) + B_f a_f \]
  \[ a = a_s(1-a_f) + a_f \]

Multi-layer Composition

- A problem when composite multiple layers:
  - The operation is not associative
  \[ (I_1, I_2)_0(I_3)_0 = (I_1, I_2)_0(I_3) \]

Revised Blending Function

- Used in Java 2D
- Proposed by Wallace & Levoy
- Algorithm:
  \[ R = R_s(1-a_f) + R_f a_f \]
  \[ G = G_s(1-a_f) + G_f a_f \]
  \[ B = B_s(1-a_f) + B_f a_f \]
  \[ a = a_s(1-a_f) + a_f \]

Premultiplied Pixel

- Color value can be pre-multiplied for efficiency
  - \( R = R / a \)
  - \( G = G / a \)
  - \( B = B / a \)
- \((r, g, b, a)\) means a pixel is \(a\)% covered by color \((r/a, g/a, b/a)\)
  - \((1/2, 0, 0, 1)\) -> half red, full coverage
  - \((1/2, 0, 0, 1/2)\) -> full red, half coverage
- Blending algorithm:
  \[ R = R_s(1-a_f) + R_f a_f \]
  \[ G = G_s(1-a_f) + G_f a_f \]
  \[ B = B_s(1-a_f) + B_f a_f \]
  \[ a = a_s(1-a_f) + a_f \]
  Be careful
  - Check whether pixel is premultiplied before do blending.