Finishing Up Assignment 1: Image Processing

COS 426: Computer Graphics (Spring 2020)
Course Logistics Update

• New course website incoming!
  – Preview at [https://reillybova.github.io/COS426-Website/](https://reillybova.github.io/COS426-Website/)
  – Should have everything, but may be slightly buggy as we work out kinks
  – If you notice any problems, please make a public Piazza post under the “website” folder

• Web Framework specs (for those interested):
  – [ReactJS](https://reactjs.org) for state-based logic and modularity
  – [MaterialUI](https://material-ui.com) to build a [Material Design](https://material.io) compliant interface
  – [GatsbyJS](https://www.gatsbyjs.org) to compile the React App to static server files (allows us to host site as a normal webpage, and makes it blazing fast)
  – Content generate from Markdown
Fill out the Assignment 0 Feedback Form

Do this **now** — it takes less than a minute:

- [https://forms.gle/o2ea1iJ978zY6Kd78](https://forms.gle/o2ea1iJ978zY6Kd78)
An Update on the Bilateral Filter

- Compute color distance in RGB space, scaled to [0, 255].

\[ w(i, j, k, l) = e^{-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|I(i, j) - I(k, l)\|^2}{2\sigma_r^2}} \]
A Familiar Pattern

Notice anything familiar about the pattern?
Why Dither?

It’s a Floyd-Steinberg dither over RGB channels (1 bit each)!

This filter was often used to compress web GIFs — look for the artifact in old-school animations!
Ordered dithering

**Pseudo code for n-bit case:**

\[
i = x \mod m \\
j = y \mod m \\
err = I(x, y) - \text{floor}_{\text{quantize}}(I(x, y)) \\
\text{threshold} = \frac{D(i, j)+ 1}{(m^2 + 1)} \\
\text{if} \ err > \text{threshold} \\
\quad P(x, y) = \text{ceil}_{\text{quantize}}(I(x, y)) \\
\text{else} \\
\quad P(x, y) = \text{floor}_{\text{quantize}}(I(x, y))
\]

- \( \text{floor}_{\text{quantize}}(p) = \frac{\text{floor}(p \times (2^n-1))}{(2^n-1)} \)
- \( \text{ceil}_{\text{quantize}}(p) = \frac{\text{ceil}(p \times (2^n-1))}{(2^n-1)} \)

\[m = 4, \ D = \begin{bmatrix} 15 & 7 & 13 & 5 \\ 3 & 11 & 1 & 9 \\ 12 & 4 & 14 & 6 \\ 0 & 8 & 2 & 10 \end{bmatrix}\]

n=1 example
Luminance
- Brightness
- Contrast
- Gamma
- Vignette
- Histogram equalization

Color
- Grayscale
- Saturation
- White balance
- Histogram matching

Filter
- Gaussian
- Sharpen
- Edge detect
- Median
- Bilateral filter

Dithering
- Quantization
- Random dithering
- Floyd-Steinberg error diffusion
- Ordered dithering

Resampling
- Bilinear sampling
- Gaussian sampling
- Translate
- Scale
- Rotate
- Swirl

Composite
- Composite
- Morph

This week's precept will focus specifically on this topic
Morph

• Basic concepts
  – transform the background image to the foreground image
  – alpha = 0: show background
  – alpha = 1: show foreground
  – alpha is the blending factor / timestamp

• General approach
  – specify correspondences (morphLines.html)
  – create an intermediate image with interpolated correspondences (alpha)
  – warp the background image to the intermediate image
  – warp the foreground image to the intermediate image
  – blend using alpha
Interpolate Morph Lines

Background Image

Foreground Image

\[
current\_line[i] = (1 - \alpha) \times background\_lines[i] + \alpha \times foreground\_lines[i]
\]
Warp Image

\[ S' = P' + u \cdot (Q' - P') \]

**Scalar**
- \( u = \frac{(X-P) \cdot (Q-P)}{||Q-P||^2} \) = Projection of PX onto PQ
- \( v = \frac{(X-P) \cdot \text{Perpendicular}(Q-P)}{||Q-P||} \) = Unit vector

**Equation**
- \( X' = P' + u \cdot (Q' - P') + \frac{v \cdot \text{Perpendicular}(Q'-P')}{||Q'-P'||} \)

**Distance**
- \( \text{dist} = \text{shortest distance from X to PQ} \)
  - \( 0 \leq u \leq 1: \text{dist} = |v| \)
  - \( u < 0: \text{dist} = ||X-P|| \)
  - \( u > 1: \text{dist} = ||X-Q|| \)

**Weight**
- \( \text{weight} = \left( \frac{\text{length}^p}{a + \text{dist}} \right) b \)
  - We use \( p = 0.5, a = 0.01, b = 2 \)

Contribution (weight) of line segment PQ to the warping of X’s location
Let \( S \) be the projection point of \( X \) onto \( PQ \)

\[
u = \frac{\text{fraction of } SP's \text{ signed length}}{	ext{over } PQ's \text{ absolute length}}
\]

\[
v = X's \text{ signed distance to } PQ, \text{ or to say, signed length of } SX \ L
\]
Warp Image

For each pixel $X$ in the destination

$DSUM = (0,0)$

$weightsum = 0$

For each line $P_iQ_i$

- calculate $u,v$ based on $P_iQ_i$
- calculate $X'_i$ based on $u,v$ and $P_iQ_i'$
- calculate displacement $D_i = X'_i - X_i$ for this line

$dist = $ shortest distance from $X$ to $P_iQ_i$

$weight = (\text{length}^P / (a + dist))^b$

$DSUM += D_i \times weight$

$weightsum += weight$

$X' = X + DSUM / weightsum$

$\text{destinationImage}(X) = \text{sourceImage}(X')$
Blending

alpha = 0.5 (also the blending factor)

Warplmage() + Warplmage()

Background Image + Foreground Image
GenerateAnimation(Image_0, L_0[...], Image_1, L_1[...])
begin
  foreach intermediate frame time t do
    for i = 1 to number of line pairs do
      L[i] = line t-th of the way from L_0[i] to L_1[i]
    end
    Warp_0 = WarpImage(Image_0, L_0, L)
    Warp_1 = WarpImage(Image_1, L_1, L)
    foreach pixel p in FinalImage do
      Result(p) = (1-t) Warp_0 + t Warp_1
    end
  end
end
Blending

alpha = 0.5 (also the blending factor)

Background Image

Foreground Image
Q&A