Floyd–Steinberg dithering

CS-116B Computer Graphics Algorithms

What is dithering?

- “Dithering is the process of juxtaposing pixels of two colors to create the illusion that a third color is present”

Source: http://webstyleguide.com/wsg2/graphics/dither.html

Why use dithering?

- Dithering makes color depth reductions less noticeable in an image for example:
- There are numerous dithering algorithms including (but not limited to):
  - Floyd–Steinberg
  - Jarvis, Judice and Ninke
  - Stucki
  - Burkes
  - Sierra
  - Atkinson

Dithering examples

Original image

Floyd-Steinberg dithering

Jarvis, Judice and Ninke dithering

Stucki dithering

Burkes dithering

Sierra dithering

Atkinson dithering

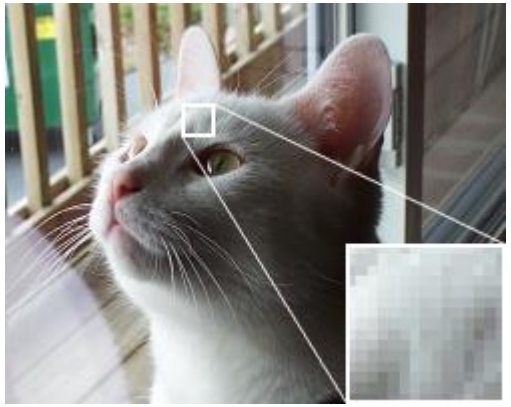
Image source: https://en.wikipedia.org/wiki/Dither

Floyd–Steinberg dithering algorithm

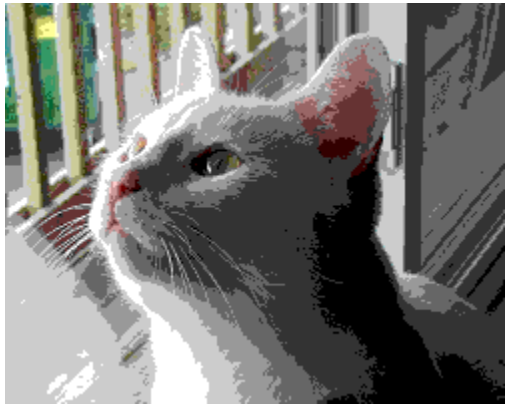
- “Diffuse the quantization error of a pixel to its neighboring pixels” (p. 16)
- “Scan in raster order” (p. 16)
- “At each pixel, draw least error output value” (p. 16)
- “Add the error fractions into adjacent, unwritten pixels” (p. 16)

Source: https://www.cs.cmu.edu/afs/cs/academic/class/15462-s09/www/lec/24/lec24.pdf

Floyd–Steinberg dithering example



Original 24-bit color image

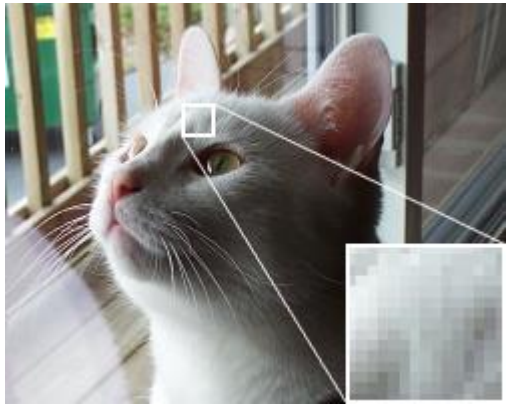


8-bit color image with no dithering

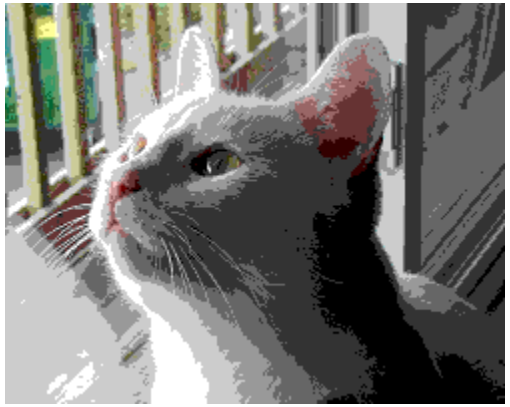
8-bit color image with Floyd-Steinberg dithering

Image source: https://en.wikipedia.org/wiki/Dither

Floyd–Steinberg dithering example



Original 24-bit color image



8-bit color image with no dithering

8-bit color image with Floyd-Steinberg dithering

Notice the diffusion of pixels and the less noticeable color-banding (posterization) between image 2 (middle) and image 3 (right) above.

Image source: https://en.wikipedia.org/wiki/Dither

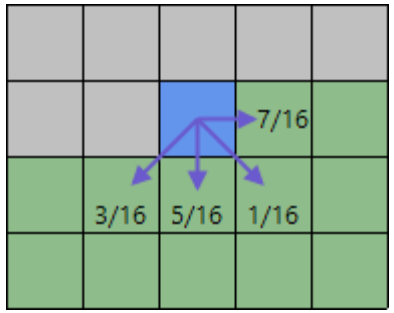
Floyd–Steinberg dithering algorithm

```
for each y from top to bottom
{
  for each x from left to right
  {
    oldpixel = pixel[x][y];
    newpixel = find_closest_palette_color (oldpixel);
    pixel[x][y] = newpixel;
    quant_error = oldpixel – newpixel;
    pixel[x+1][y] = pixel[x+1][y] + quant_error * 7/16;
    pixel[x-1][y+1] = pixel[x-1][y+1] + quant_error * 3/16;
    pixel[x][y+1] = pixel[x][y+1] + quant_error * 5/16;
    pixel[x+1][y+1] = pixel[x+1][y+1] + quant_error * 1/16;
  }
}
```

Source: https://www.cs.cmu.edu/afs/cs/academic/class/15462-s09/www/lec/24/lec24.pdf

Floyd–Steinberg dithering algorithm

```
for each y from top to bottom
{
  for each x from left to right
  {
    oldpixel = pixel[x][y];
    newpixel = find_closest_palette_color (oldpixel);
    pixel[x][y] = newpixel;
    quant_error = oldpixel – newpixel;
    pixel[x+1][y] = pixel[x+1][y] + quant_error * 7/16;
    pixel[x-1][y+1] = pixel[x-1][y+1] + quant_error * 3/16;
    pixel[x][y+1] = pixel[x][y+1] + quant_error * 5/16;
    pixel[x+1][y+1] = pixel[x+1][y+1] + quant_error * 1/16;
  }
}
```



Sources: https://www.cs.cmu.edu/afs/cs/academic/class/15462-s09/www/lec/24/lec24.pdf and https://www.cyotek.com/files/articleimages/dithering-floyd-steinberg-diagram.png?version=12840

For Further Reading

Crocker, L. D., Boulay, P., & Morra, M. (1991). DHALF.TXT. http://www.efg2.com/Lab/Library/ImageProcessing/DHALF.TXT

CS 559: Computer Graphics: Floyd-Steinberg Dithering. https://research.cs.wisc.edu/graphics/Courses/559-s2004/docs/floyd-steinberg.pdf

Floyd, R. W., & Steinberg, L. (1976). An adaptive algorithm for spatial grey scale. *Proceedings of the Society of Information Display 17*, 75–77.

Helland, T. (n. d.). http://www.tannerhelland.com/4660/dithering-eleven-algorithms-source-code/

Moss, R. J. (2015). Dithering an Image Using the Floyd-Steinberg Algorithm in C#. https://www.codeproject.com/Articles/1001088/Dithering-an-Image-Using-the-Floyd-Steinberg-Algor