Assignment — Image Processing

CSCI 4471.2 – Computer Graphics
Dr. Sageev Oore
January 2015

1 Creating a set of images for you to play with...

Start by creating the following images which will form the bases of your experiments. I recommend you start with greyscale images (where the red, blue and green components are all equal to each other), although later you could explore colour effects as well.

- *Constant*: Set each pixel to a given constant intensity value.
- *White noise*: Set each pixel to a random float between 0 (dark) and 1 (bright), independent of every other pixel
- *An interesting noise... :*
  - Step 0: Set all pixel values to 0.5
  - Step 1: Randomly choose 2 pixels in the image; note that they define a straight line
  - Step 2: For all pixels to the left of the line, increase their brightness by δ; for all pixels to the right of the line, decrease their brightness by δ.
  - Go back to step 1.

  Play around with different values of δ. What happens if you choose δ randomly on each iteration? What happens if you apply δ multiplicatively versus additively?
- *Photographs*: Load in a few photographs of your choice
- *Gaussian*: Let $\mathbf{u}$ represent the centre pixel of the image. Let $\mathbf{p}$ represent any pixel in the image. Define the brightness at $\mathbf{p}$ to be

$$value(\mathbf{p}) = \exp\left\{\frac{-\|\mathbf{p}-\mathbf{u}\|^2}{2\sigma^2}\right\}$$

  .
- *Stripes*: Create an image filled with parallel lines. Can your stripe function take a parameter θ so that the stripes are at an orientation θ?

For each of the above images, create some "large" versions (e.g. 300x300 or more) and some "tiny" versions (e.g. 3x3, 10x10).

2 Single-image manipulation

2.1 Brightness

Write a function B such that if I is an input image, then $B(I)$ returns the total brightness in the image, i.e.

$$B(I) = \sum_{pixel p \in I} p.value$$

where $p.value$ is the greyscale value (i.e. intensity[1]) of pixel p.

2.2 Equalize

Write a function E that transforms the pixel values such that the minimum pixel intensity is 0 and the maximum is 1. This is an essential function.

2.3 Normalization

Write a function N such that if I_1 is an input image, then the resulting image $I_2 = N(I_1)$ has brightness $B(I_2) = 1$.

2.4 Histogram

Write a function $H(I, n)$ that takes in an image I and an integer n, and returns an array h of length n as follows:

- partition the interval $[0, 1)$ into n equal subintervals, e.g. $[0, 1/n), [1/n, 2/n), \ldots, [(n-1)/n, 1)$.
- for each subinterval $[a, b)$, count the number of pixels whose intensity lies in that interval. That is the value in the corresponding array element. For example, if $n = 2$, and there are 10 pixels with intensity less than 0.5, then $h[0] = 10$. You can also write this as

$$h[i] = \sum_{p.value \in [a_i, b_i)} 1$$

Note the above computations assume the intensities are in the interval $[0, 1)$. If the intensities are actually integers in the range $[0, 255]$, as they most likely are, then you can of course simply approximate this by dividing each intensity by 256.

If the bins contain very different numbers of pixels, is there some way you can modify some or all of the pixel values so that they become more evenly distributed among the bins? This is sometimes known as histogram equalization.

[1] For now we are acting as though intensity is the same as brightness. This is a technically incorrect use of these terms.

2.5 Quantization

Write a function $Q(I, n)$ that takes in an image I and an integer n and returns a new image that has only n distinct intensity values. For example, $Q(I, 2)$ returns a black & white image, whereas $Q(I, 256)$ returns the same image as the input image (assuming the input image had 8-bit intensity values).

There are different ways of quantizing an image. For example, you can compute an n-bin histogram, and then set all the pixels in each bin to the centre value of the bin. Or, you could set all the pixels in each bin to the average value of the intensities of all those pixels in that bin. Or, you could start by sorting the pixels in the image according to their intensity, and partition this list into n bins so that each bin ends up containing a roughly similar number of pixels. Play around... Try to understand and explain how the different quantization algorithms lead to different resulting quantized images.

2.6 Resizing

How can you make an image twice its size? How can you make it half its size? There are a variety of ways to do this.

3 Multi-Image Manipulation

3.1 Linear Combinations

Try taking linear combinations of two equally-sized images, e.g. $I_1 - I_2$, $I_1 + I_2$, $\alpha I_1 + (1 - \alpha) I_2$. Note that you will often need to normalize or equalize the resulting image in some way. What if your weighting parameter α changes across the image, e.g. $\alpha = 0$ along the left edge of the image, $\alpha = 1$ along the right edge, and varies smoothy in between the two borders?

3.2 Stitching

How can you take two images and join them together as smoothly as possible (e.g. to create a panorama). (Hard!)

3.3 Convolution

Take a large and a tiny image and convolve them. Tons of stuff to try here. Here are just some initial ideas:

- Try convolving various pairs of images from the list you created above. Let T be a set of tiny images that include one or more white noises, a few gaussians (with different values of σ), a couple of constant images, a few differently oriented stripes. Let L be a set of large images, including several photographs, some stripes, white noise. Convolve each of the tiny images with each of the large images.

- Let $W(I_1, I_2, \alpha) = \alpha I_1 + (1 - \alpha) I_2$ be a weighted average of two images I_1 and I_2. Create a few different large and tiny grids by averaging (and possibly equalizing) pairs of stripes. Convolve a set of stripes with a grid.
- Create "soft" stripes by convolving a tiny gaussian with a large stripes. Now convolve the soft stripes with some photographs.
- What happens when you convolve tiny white noise with stripes? What about tiny white noise with large white noise?
- Can you transform a tiny gaussian (say 20x20) so that it is in the shape of a diagonally pointing ellipse rather than a symmetric circle? What happens if you convolve this with some large images?
- a tiny gaussian
- Using the functions you implemented earlier, let $DG = N(E(N(G_1) - N(G_2)))$, i.e. create two normalized gaussians, subtract one from the other, and renormalize. DG represents a difference of gaussians. Now convolve DG with some photographic images.

There are many more things to try. It is very important that you not only play around with different algorithms and parameters in these exercises, but also that you try to analyze and explain the results. Sometimes, in order to understand something, you might need to come up with an additional set of example images.