Re: 'Magic' kernel for image zoom resampling?

Source: http://sci.tech−archive.net/Archive/sci.image.processing/2006−06/msg00057.html

- *From*: jim <"sjedgingN0sp"@m@xxxxxxx>
- *Date*: Thu, 08 Jun 2006 08:08:39 −0500

Hello John,
I am pretty sure you are not the first to use this kernel, altho many people avoid even lehgth kernels so they would never use it.

In an ideal world to downsample by 2 you would first filter by a perfect halfband filter and then discard every other sample. It appears that you have settled on a filter with 1D length 4 and comprised of integer coefficients. Given those 2 constraints I would say there is a reasonable argument that the coefficients you have selected are as close to the perfect halfband filter as you can possibly get. They may even be as close as you can get even without the constraint that the coefficients be integers.

If you were to allow larger kernels then length 4 I think you could do better. Then a 1D kernel of [−1 0 3 4 3 0 −1]/4 would be considerably closer to the perfect halfband response then the one you have chosen. Notice that only has one additional coefficient to multiply and the normalizing factor is still a power of 2. Of course some folks will not like the effects created by negative coefficients, but you can't please everyone.

By the way, I don't fully understand the explanation of the upsampling technique. Is it equivalent to inserting a zero at every other sample and then filtering with your magic kernel or are you doing something completely different?

−jim

jpcostella@xxxxxxxxxxx wrote:

> Hi folks,
>
> I've recently generated a solution to an image zooming problem, and can't find anything quite like it on our good old web or usenet. I'm

trying to figure out if I've stumbled on something that might actually be useful to other people. :)

My need was to downsample an image by a factor of 8 in each direction (i.e. 1 downsampled pixel for every 64 original image pixels), and then upsample the result back to the original resolution. I understand all the information theoretical aspects of this process; I was trying to figure out if there was a kernel that did both tasks well (i.e., the final result was smooth and without stellations or other strange artifacts), and also did them fast.

After some truly terrible attempts (even though I thought I had sinc functions well and truly under my belt 20 years ago), I found that the following recursive algorithm works amazingly well:

1. To downsample or upsample by a factor of 2, use a kernel

1 3 3 1
3 9 9 3
3 9 9 3
1 3 3 1

placed over every second pixel in each direction. (Total normalization of 64 is relevant for downsampling. For upsampling, only divide by 16, because every upsampled pixel gets a contribution from four downsampled pixels' kernels).

2. To downsample or upsample by a factor of 2^k, perform the above resampling recursively, k times.

The results for upsampling are astoundingly good, really astoundingly good, and better than what I could get Photoshop to do. (Some years ago I did my fair share of 'extreme zooming' in some work on the photographic evidence of the JFK assassination, so I'm very familiar with all the strange artifacts that can pop out, and the disappointment that accompanies it.)

I can upload a few images to my website if need be, to demonstrate what I am talking about (or, equivalently, point me to some test images and I'll show you what comes out).

For my particular purpose (8:8 downsampling and upsampling), applying this 'magic' kernel three times yields a kernel that is only 22 x 22 in size (if you want to precompute the kernel and apply it once, as I ultimately want to, rather than actually perfoming the doubling process three times). (Every time you apply it, double the kernel width or height and add 2, so 2 x 4 + 2 = 10, and 2 x 10 + 2 = 22.) That's pretty good, considering that, for 800% zooming, it's already 16 pixels from nearest pixel to nearest pixel, in each dimension.

Of course, if you wanted to resample by something that is not a power

of 2, then you'd need to use the 'magic' kernel for the nearest power of 2, and then use something more traditional for the final small adjustment in resolution. That's no major problem, because getting to the final result from the nearest power of 2 is never worse than between a 71% and 141% zoom, and just about any resampling method does a decent job in this range.

My question is: Is this 'magic kernel' something new, or is this trick known?

The closest I could find on the net is the 'stair interpolation' trick, which uses Photoshop's bicubic for successive 110% increases, which is sort of, kind of, the same idea, but not quite. The other resampling kernels I could find on the net look much more like what I was trying to do in the first place, but nothing like what I ended up with.

The 'magic' kernel sort of reminds me of the Fast Fourier Transform, which also gets all sorts of amazing efficiencies with powers of 2, and then needs a bit of a (non−critical) fudge if you don't quite have a power of 2.

Oh, and by the way, I know how I arrived at the 'magic' kernel above (for another aspect of the project that needed just 2:2 downsampling and upsampling), and it has some nice mathematical properties (it is 'separable' in the sense that it is the product of an 'x' 1−3−3−1 and a 'y' 1−3−3−1, and its normalization is always a power of 2, which means everything is integer look−ups with bit−shifts, which makes me extremely happy), but I have no mathematical proof at all of why it works so damn well when applied to itself recursively.

Anyhow, thanks in advance for any words of advice. And apologies in advance if I've rediscovered someone's magic kernel, as is bound to be the case. :)

John Costella

Dr. John P. Costella
BE(Elec)(Hons) BSc(Hons) PhD(Physics) GradDipEd
Melbourne, Victoria, Australia
jpcostella@xxxxxxxxxxx
costella@xxxxxxxxxxx
assassinationscience.com/johncostella

.