

# What is the idea behind scaling an image using Lanczos?

Asked 15 years ago   Modified 3 years, 10 months ago   Viewed 45k times

---



40



I'm interested in image scaling algorithms and have implemented the bilinear and bicubic methods. However, I have heard of the Lanczos and other more sophisticated methods for even higher quality image scaling, and I am very curious how they work.



Could someone here explain the basic idea behind scaling an image using Lanczos (both upscaling and downscaling) and why it results in higher quality?

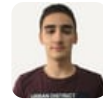
I do have a background in Fourier analysis and have done some signal processing stuff in the past, but not with relation to image processing, so don't be afraid to use terms like "frequency response" and such in your answer :)

EDIT: I guess what I really want to know is the concept and theory behind using a convolution filter for interpolation.

(Note: I have already read the Wikipedia article on Lanczos resampling but it didn't have nearly enough detail for me)

[algorithm](#)[image](#)[graphics](#)[image-processing](#)[image-scaling](#)[Share](#)[Improve this question](#)[Follow](#)

edited Feb 18, 2021 at 14:09

[Vukašin Manojlović](#)

3,857 ● 3 ● 22 ● 33

asked Dec 6, 2009 at 2:27

[horseyguy](#)

29.9k ● 20 ● 108 ● 147

---

have you looked at any example code? I don't know how readable e.g. mplayer's swscale implementation is. But you're probably looking for the theory behind it, which I don't know either. – [Peter Cordes](#) Dec 6, 2009 at 8:41

---

- 5 Interested readers will find an illustrated, extended examination of this question on our sister site at [gis.stackexchange.com/a/14361](https://gis.stackexchange.com/a/14361) – [whuber](#) May 7, 2012 at 21:28
- 

There's an excellent article [comparing resampling algorithms](#), although its test is rotation, rather than scaling.  
– [Roman Starkov](#) Mar 4, 2013 at 12:51

---

The Catmull-Rom bicubic formula produces results that are very close to a Lanczos-2, but the math is much simpler.  
– [Mark Ransom](#) Oct 28, 2016 at 17:44

---

[icess.eri.ucsb.edu/gem/Duchon\\_1979\\_JAM\\_Lanczos.pdf](https://icess.eri.ucsb.edu/gem/Duchon_1979_JAM_Lanczos.pdf)  
– [swiss\\_knight](#) Mar 22, 2021 at 18:51

---

# 1 Answer

Sorted by:

Highest score (default)



50



The selection of a particular filter for image processing is something of a black art, because the main criterion for judging the result is subjective: in computer graphics, the ultimate question is almost always: "does it look good?".

There are a lot of good filters out there, and the choice between the best frequently comes down to a judgement call.

That said, I will go ahead with some theory...

---

Since you are familiar with Fourier analysis for signal processing, you don't really need to know much more to apply it to image processing -- all the filters of immediate interest are "separable", which basically means you can apply them independently in the x and y directions. This reduces the problem of resampling a (2-D) image to the problem of resampling a (1-D) signal. Instead of a function of time (t), your signal is a function of one of the coordinate axes (say, x); everything else is exactly the same.

Ultimately, the reason you need to use a filter at all is to avoid aliasing: if you are reducing the resolution, you need to filter out high-frequency original data that the new, lower resolution doesn't support, or it will be added to unrelated frequencies instead.

So. While you're filtering out unwanted frequencies from the original, you want to preserve as much of the original signal as you can. Also, you don't want to distort the signal you do preserve. Finally, you want to extinguish the unwanted frequencies as completely as possible. This means -- in theory -- that a good filter should be a "box" function in frequency space: with zero response for frequencies above the cutoff, unity response for frequencies below the cutoff, and a step function in between. And, in theory, this response is achievable: as you may know, a straight sinc filter will give you exactly that.

---

There are two problems with this. First, a straight sinc filter is unbounded, and doesn't drop off very fast; this means that doing a straightforward convolution will be very slow. Rather than direct convolution, it is faster to use an FFT and do the filtering in frequency space...

However, if you actually do use a straight sinc filter, the problem is that it doesn't actually look very good! As the related question says, perceptually there are ringing artifacts, and practically there is no completely satisfactory way to deal with the negative values that result from "undershoot".

Finally, then: one way to deal with the problem is to start out with a sinc filter (for its good theoretical properties), and tweak it until you have something that also solves your other problems. Specifically, this will get you something like the Lanczos filter:

```
Lanczos filter:      L(x)      = sinc(pi x)
sinc(pi x/a) box(|x|<a)
frequency response: F[L(x)](f) = box(|f|<1/2) *
box(|f|<1/2a) * sinc(2 pi f a)
```

[note that "\*" here is convolution, not multiplication]

[also, I am ignoring normalization completely...]


- the  $\text{sinc}(\pi x)$  determines the overall shape of the frequency response (for larger  $a$ , the frequency response looks more and more like a box function)
- the  $\text{box}(|x|<a)$  gives it finite support, so you can use direct convolution
- the  $\text{sinc}(\pi x/a)$  smooths out the edges of the box and (consequently? equivalently?) greatly improves the rejection of undesirable high frequencies
- the last two factors ("the window") also tone down the ringing; they make a vast improvement in both the perceptual artifact and the practical incidence of "undershoot" -- though without completely eliminating them

Please note that there is no magic about any of this. There are a wide variety of windows available, which work just about as well. Also, for  $a=1$  and  $2$ , the frequency response does not look much like a step function. However, I hope this answers your question "why sinc", and gives you some idea about frequency responses and so forth.

answered Dec 7, 2009 at 8:34

[comingstorm](#)

26.1k ● 3 ● 46 ● 68

- 
- 1 I know this answer is very old by now, but I do wonder: Is it really desirable to downscale images with a frequency box filter? Wouldn't that intrinsically mean that such things as one-pixel outlines are eliminated (by the ideal filter)? Is that ever desirable, or are sinc-based filters only really good for images that can be considered "continuous" in nature (such as raster-sampled photographs), and never for images that are more pixel-arty in nature? – [Dolda2000](#) Dec 9, 2012 at 6:43
- 
- 4 One-pixel outlines would not disappear, but they do yield a particularly clear version of the visually obnoxious "ringing" effect described above when put through a pure sinc filter. Lanczos-style filters are better in this respect, but you are correct that sinc-based filters are most appropriate for "continuous" images -- pixel-art-style images don't do well with *any* kind of linear resizing filter, due to their reliance on sharp horizontal and vertical edges. – [comingstorm](#) Feb 13, 2013 at 19:01 
-