Digital Filtering Tutorial for Computer Graphics

Alvy Ray Smith
Computer Graphics Project
Lucasfilm Ltd

Technical Memo No. 27
20 Nov 1981 (Revised 3 Mar 1983)

Presented as tutorial notes at the 1983 SIGGRAPH, July 1983, and the 1984 SIGGRAPH, July 1984. This document was reentered in Microsoft Word on 8 Feb 2000. Spelling and punctuation are generally preserved, but trivially minor spelling errors are corrected. Otherwise additions or changes made to the original are noted inside square brackets or in footnotes.

Digital sampling and filtering in both space and time are intrinsic to computer graphics. The pixels of a framebuffer representation of a picture are regularly placed samples in 2-dimensional screen space. Inappropriate application of sampling theory (or no application at all) results in the artifact called "jaggies". The frames of a film representation of a movement are regularly placed samples in time. The artifact here is called "strobing". Spatial filtering, called "antialiasing", is used to soften the jaggies. Temporal filtering, called "motion blur", removes strobing of edges and backward spinning stagecoach wheels.

The purpose of this memo is to review the principles of digital sampling and filtering theory in the context of computer graphics. In particular, it is a rewording of the classical results in terms with which I am comfortable. Hopefully other computer graphicists will also find the restatement helpful.

We will deal here only with the spatial case. The two examples studied are scaling a picture up *(magnification)* and scaling a picture down *(minification)*. We shall be especially concerned with what happens as magnification becomes minification—as the scale factor passes through 1. We begin with well-known theorems and derive the principal result: four equivalent statements of the minification process and four equivalent statements of the magnification process.

Everyone seems to believe that sampling theory is simple, elegant, and straightforward, but it is my experience that whenever two computer graphicists try to discuss the subject they end up quibbling over details, sacrificing intuition in the impossible effort to convince one another that each knows the important details of the basic theory. I think this is because there are several (at least four) ways to look at a sampling process, all equivalent, and each combatant is fluent in only one or two of these ways and never the same ones as his opponent.

Not only do I hope to offer a set of wordings to form a basis for comfortable conversation but also a set of equivalent intuitions among which we can pick and choose depending on the particular constraints of the hardware of software design problem at hand.

I warn the reader that the math is anything but rigorous. I omit normalization factors, for example, and the spectra in the figures are not accurate transformations of the accompanying spatial functions. I concentrate instead on intuition-forming diagrams and assume that the mathematical details can be readily filled in once the underlying concepts are fully grasped.

BASICS

I shall assume the reader is at home with the following basic notions:

(1) Nyquist or sampling, theorem. A real-valued function bandlimited to frequency f can be perfectly represented by a set of equally-spaced samples of the function taken at a rate of $2f$. Furthermore, the real function can be recovered from the sampled version by convolving the samples with an appropriate filter (the sinc(x) function). See Fig. 1 for the sinc(x) function and its shape relative the sampling interval. Inadequately sampled functions, or insufficiently low-pass filtered functions, exhibit "aliasing" which manifests spatially as jaggies or picture breakup (visible in a sequence of improperly sampled pictures).

(2) The Fourier transform. The transform of the sinc function is a box in the frequency domain (a low-pass filter). If $G(f)$ is the transform of a real function $g(x)$, then the transform of the sampled version of $g(x)$ is an infinite sequence of translations of $G(f)$. See Fig. 2.

(3) Scaling a function's abscissa. I shall refer to scaling a function up or down by which I will mean scaling its abscissa, not its ordinate. If $g(x)$ is a given function, then if it is scaled down by $1/a, a > 1.$, the new function is $g(ax)$. If its scaled up by $a > 1.$, then the new function is $g(x/a)$. See Fig. 3 for the graphical meaning of scaling up and down. Also indicated in this figure is the fact that scaling down (up) a function of x is equivalent to scaling up (down) its frequency content.

(4) Convolution. Convolution in the space domain is equivalent to multiplication in the frequency. The convultion of $g(x)$ with $h(x)$ is also a function of x which I shall represent by

$$c(x) = [g(x) * h(x)](x).$$

Note in particular that if this convolution is scaled by the real factor a, then the scaled convolution is given by

$$c(ax) = [g(x) * h(x)](ax)..$$

(5) Specifically, we will use the fact that convolution with sinc(x) in space is equivalent to low-pass filtering the frequency spectrum with a box—ie, multiplying it by box(f).

REASONING FROM BASICS: MINIFICATION

Minification, shrinking, scaling down, decimation, and sampling rate reduction are all terms used to describe one process. In terms of pictures, when we reduce the size of the picture we minify it (cf, magnify for the opposite process). I will describe this process in as straightforward a manner as I can using basic concepts. I will do the same for magnification in the next section. This will lead to a very simple set of descriptions for the two processes in digital filtering terms.

Finally, by application of two simple theorems, this pair of descriptions will be converted into three other equivalent pairs of descriptions.

In computer graphics terms, the problem of minification is to convert a sampled version of a picture into a sampled version of the same picture reduced in size. This is to be accomplished with as much integrity as possible and without introduction of aliasing. We assume that $g(x)$ is the original picture and that $\hat{g}(x)$ is the sampled version (residing, for example, in the pixels of a framebuffer). In other words, what we are given is $\hat{g}(x)$. For convenience, only the 1-dimensional case is studied here. Note that this is completely sufficient for 2-pass transforms of pictures [1]. We assume that $g(x)$ can be accurately reconstructed from the given samples $\hat{g}(x)$—ie, that $g(x)$ has been correctly low-pass filtered.

What we want to obtain is the set of samples which correctly represents a scaled down version of $g(x)$. Let $h(x) = g(ax), a > 1$. Thus h is g reduced by a factor $1/a$. What we want is $\hat{h}(x)$, the correctly sampled representation of $h(x)$. These samples are what we would write into a framebuffer, say. There are two ways to do this but one is substantially easier to compute than the other.

The Hard Way

(1) Reconstruct $g(x)$ from the given $\hat{g}(x)$ samples by convolution.

$$g(x) = [\hat{g}(x) * \mathrm{sinc}(x)](x).$$

This is shown in Fig. 4a-4b in both space and frequency domains.

(2) Scale $g(x)$ to obtain $g'(x)$.

$$g'(x) = g(ax) = [\hat{g}(x) * \mathrm{sinc}(x)](ax).$$

This stretches $G(\mathrm{f})$, the Fourier transform of $g(x)$, by a in the f direction to obtain $G'(f)$, the transform of $g'(x)$. See Fig. 4c.

(3) Since, by scaling down, high frequencies may have been introduced into the function, we must low-pass filter the function before resampling.

$$h(x) = [g'(x) * \mathrm{sinc}(x)](x) = [[\hat{g}(x) * \mathrm{sinc}(x)](ax) * \mathrm{sinc}(x)](x).$$

In the frequency domain, $G'(f)$ is low-pass filtered to obtain $H(f)$. See Fig. 4d.

(4) $\hat{h}(x)$ is simply $h(x)$ at the pixels. Notice that we assume here that the sample points remain fixed and the functions change shape relative to these fixed points. This point is key to our various equivalent models derived later.

The expression above for $h(x)$ is quite difficult. Fortunately, there is a simpler expression which can be derived using Fourier transform theorems.

The Easy Way

See Fig. 5a for a summary of what we just did in the frequency domain. The replicated spectrum of $g(x)$, due to the sampling process, was reduced to a single copy of the spectrum by low-pass filtering (convolving with $\mathrm{sinc}(x)$ in the space domain). Then $G(f)$ was stretched to get $G'(f)$, which was finally low-pass filtered to get $H(f)$. Fig. 5b shows that we would obtain the same result if we skip the first step, the reconstruction of $g(x)$ from $\hat{g}(x)$. If we simply stretch the spectrum $\hat{G}(f)$ instead of $G(f)$, then low-pass filter the result $G'(f)$, we would obtain the same result $H(f)$ as in Fig. 5a. In the space domain this second process translates into:

(1) Scale $\hat{g}(x)$ by $1/a$:

$$\hat{g}'(x) = \hat{g}(ax).$$

(2) Get rid of added high frequencies:

$$h(x) = [\hat{g}(ax) * \text{sinc}(x)](x).$$

(3) $\hat{h}(x)$ is $h(x)$ at the pixels.

Fig. 6 is a graphical summary of this easier method of obtaining the desired output pixels. This simplification is also easy to prove by looking in frequency f domain where

$$\text{box}(mf)\text{box}(nf) = \text{box}(\min(m,n)\,f).$$

Thus

$$\text{sinc}(ax) * \text{sinc}(x) = \text{sinc}(x)$$

for $a > 1$.

REASONING FROM BASICS: MAGNIFICATION

Magnification, stretching, scaling up, interpolation, and sampling rate increase are all terms used to describe one process. In terms of pictures, when we increase the size of the picture, we magnify it. Magnification does not require any computation-reducing manipulations, as did minification. It is basically straightforward.

(1) Reconstruct $g(x)$ from the given samples $\hat{g}(x)$

$$g(x) = [\hat{g}(x) * \text{sinc}(x)](x).$$

This process in both space and frequency is illustrated in Fig. 7.

(2) Scale $g(x)$ up by $a > 1$.

$$g'(x) = g(x/a).$$

(3) Since scaling up reduces frequency content, we don't have to low-pass filter as we did in the minification case before sampling. That is, $h(x) = g'(x)$ and

$$\hat{h}(x) = [\hat{g}(x) * \text{sinc}(x)](x/a)$$

at the pixels.

Fig. 8 summarizes the filtering implied by this math.

MAGNIFY/MINIFY: FORMULATION A

A summary description of the two filtering processes just detailed is given below in a form convenient for our purposes. By "attaching a weighted filter to a pixel", I mean that the filter is centered at the position of the pixel and its scaled by the height of the pixel.

Magnify:

(a) Attach magnified weighted filters to each input pixel *image*.
(b) Sum contributions at each output pixel.

Minify:

(a) Attach weighted filters to each input pixel *image*.
(b) Sum contributions at each output pixel.

In the magnify case, the filter is magnified by the amount that $g(x)$ is magnified to get $g'(x)$. If $g'(\mathrm{x}) = \mathrm{g}(x/a)$ and the filter function is $f(x)$, then $f'(x) = \mathrm{f}(x/a), a > 1$. (We will use f to represent both the filter domain variable and an arbitrary filter function; the context will make clear which is intended.)

A THEOREM

This easily derived theorem just states the fact that scaling a convolution of two functions is equivalent to convolving the two functions scaled.

Theorem 1. If the convolution of $g(x)$ with $h(x)$ is given by
$$c(x) = [g(x) * h(x)](x),$$
then
$$c(ax) = [g(x) * h(x)](ax) = [g(ax) * h(ax)](x).$$

MAGNIFY/MINIFY: FORMULATION B

Fig. 9 shows that by a simple scaling all the manipulations performed in Formulation A above in the input domain can be equivalently carried out in the output domain and vice versa. This is simply an application of Theorem 1 to Formulation A. Thus our second formulation is as follows:

Magnify:
(a) Attach weighted filters to each input pixel.
(b) Sum contributions at each output pixel *preimage*.

Minify:
(a) Attach magnified weighted filters to each input pixel.
(b) Sum contributions at each output pixel *preimage*.

In the minify case, the filter is magnified by the amount that $g'(x)$ would have to be magnified to obtain $g(x)$. If $g'(x) = g(ax)$, then $f'(x) = f(x/a), a > 1$

ANOTHER THEOREM

The following easily derived theorem gives us two more formulations:

Theorem 2. For *symmetric filters*, the sum at location x of weighted filters spaced at distance T is equal to the sum of samples of a single unweighted filter positioned at x where filter samples are T apart and weighted by the sample at the filter sample point.

Fig. 10 illustrates the Theorem. The sum in Fig. 10a at location x_0 is
$$Af(x_0 - a) + Bf(x_0 - b) + Cf(x_0 - c) + Df(x_0 - d).$$
For Fig. 10b it is
$$Af(a - x_0) + Bf(b - x_0) + Cf(c - x_0) + Df(d - x_0).$$

These are clearly the same if $f(x) = f(-x)$—ie, if f is a symmetric filter. The theorem actually holds for asymmetric filters also, using the same proof technique as above. The generalization requires only a slight rewording to use the negative of the filter.

Theorem 2 (Generalized). For an arbitrary filter $f(x)$, the sum at location x of weighted filters f spaced at distances T from x is equal to the sum of samples of a single unweighted filter $f'(x) = f(-x)$ positioned at x where filter samples are taken at distances T from x and weighted by the sample at the filter sample point.

Since we usually deal with symmetric filters (except in the case of perspective [cf, item (4) in the Conclusion]), at the first version of Theorem 2 will suffice.

MAGNIFY/MINIFY: FORMULATION C

This formulation is obtained by applying Theorem 2 to Formulation A.

Magnify:

(a) Position magnified unweighted filter at each output pixel.
(b) Sum this filter's samples at each input pixel image under it, weighted by the corresponding input pixel.

Minify:

(a) Position unweighted filter at each output pixel.
(b) Sum this filter's samples at each input pixel image under it, weighted by the corresponding input pixel.

MAGNIFY/MINIFY: FORMULATION D

This formulation is obtained by applying Theorem 2 to Formulation B.

Magnify:

(a) Position unweighted filter at each output pixel preimage.
(b) Sum this filter's samples at each input pixel under it, weighted by the corresponding input pixel.

Minify:

(a) Position magnified unweighted filter at each output pixel preimage.
(b) Sum this filter's samples at each input pixel under it, weighted by the corresponding input pixel.

CONCLUSION

We can choose any formulation for magnify from the four above and can independently choose any for minify. So let's choose a pair which seems particularly convenient, Magnify Formulation C and Minify Formulation C—ie, Formulation C:

Magnify:

(a) Position magnified unweighted filter at each output pixel.
(b) Sum this filter's samples at each input pixel *image* under it, weighted by the corresponding input pixel.

Minify:

(a) Position unweighted filter at each output pixel.
(b) Sum this filter's samples at each input pixel *image* under it, weighted by the corresponding input pixel.

The result then is that *to go smoothly from minify to magnify we must simply stretch the filter as we begin to magnify*. But stretching a filter is simply scaling down the index into it (assuming it is represented by a finely divided table). Fig. 11 shows how a filter would be affected as the scale factor changes.

We can put Fig. 11 into words: Let $P(x)$ be the intensity of the input pixel at input pixel location x. Let x' be the image of x, and let x'_L be the image of x_L which is the input pixel location just left of x—ie, $x - 1$. Then let a be the scaling factor—ie,

$$a = |x' - x'_L|.$$

Finally, let x_{out} be the output pixel location for which we are calculating the intensity, and $f(x)$ be the filter function. Then we have the following:

Magnify $(a > 1)$**:**

The value which the input pixel at x contributes to the output pixel at x_{out} is the amount

$$f\left(\frac{x' - x_{out}}{a}\right) * P(x)$$

(ie, the index into f(x) is scaled down by $1/a$).

Minify $(a < 1)$**:**

The value which the input pixel at x contributes to the output pixel at x_{out} is the amount

$$f(x' - x_{out}) * P(x).$$

Consider a filter of support 4—ie, it extends plus or minus 2 pixels from the pixel it is centered on. It is interesting to notice that, for magnification, there are always exactly 4 input pixel images under the (unweighted stretched) filter.

Issues we have not covered are

(1) Normalization to ensure that flat fields remain flat at the same value. The trick is to filter table entries as well as weighted table entries (weighted by the intensity of the corresponding pixels), then divide the weighted sum by the unweighted sum.
(2) The effect of finiteness of both spatial and frequency extent on the theory above. This is dealt with in FFT literature.
(3) The choice of a good filter, assuming a finite filter is desired (eg, one of support 4). There is a mountain of literature on this subject [2, 3].
(4) Variable rate sampling (as occurs in perspective mappings) and its effect on the theory above.

REFERENCES

[1] Edwin Catmull and Alvy Ray Smith, *3-D Transformations of Images in Scanline Order*, **Computer Graphics,** Vol 14, No 3, 279-285, Jul, 1980. (SIGGRAPH 80 Conference Proceedings).

[2] Alan V Oppenheim and Ronald W Schafer, **Digital Signal Processing,** Prentice-Hall, Inc, Englewood Cliffs, NJ, 1975.

[3] Lawrence R Rabiner and Bernard Gold, **Theory and Application of Digital Signal Processing** Prentice-Hall, Inc, Englewood Cliffs, NJ, 1975.

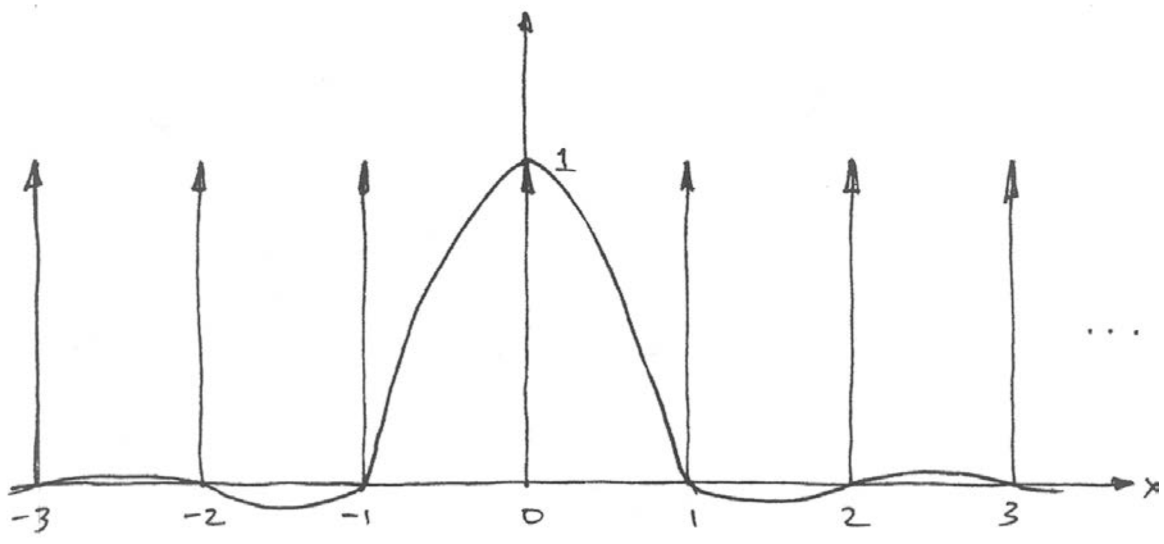
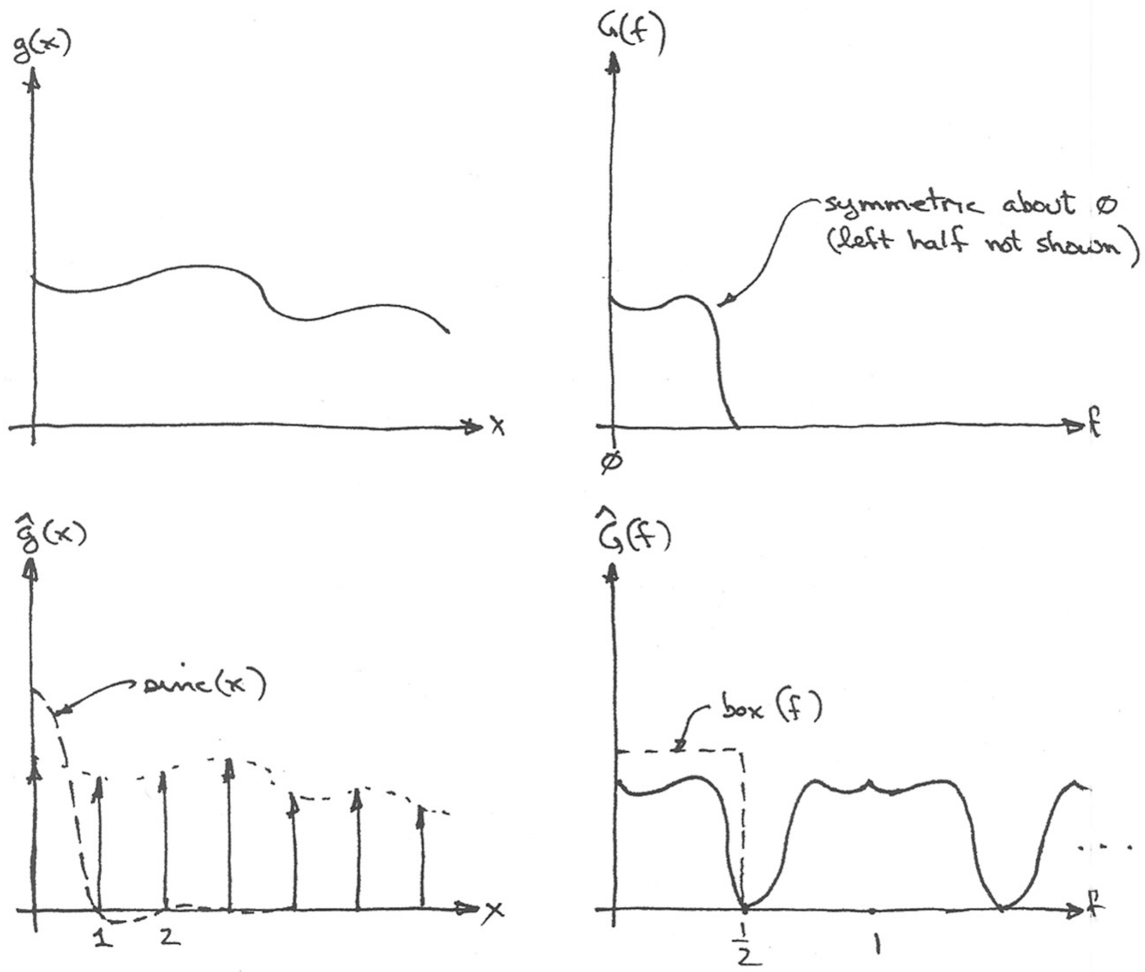


Fig. 1. Pixel rate sampling function and the corresonding reconstruction filter, sinc(x) = sin(πx)/(πx).

Fig. 2. A sampled function's spectrum is an infinite sequence of the unsampled function's spectrum.



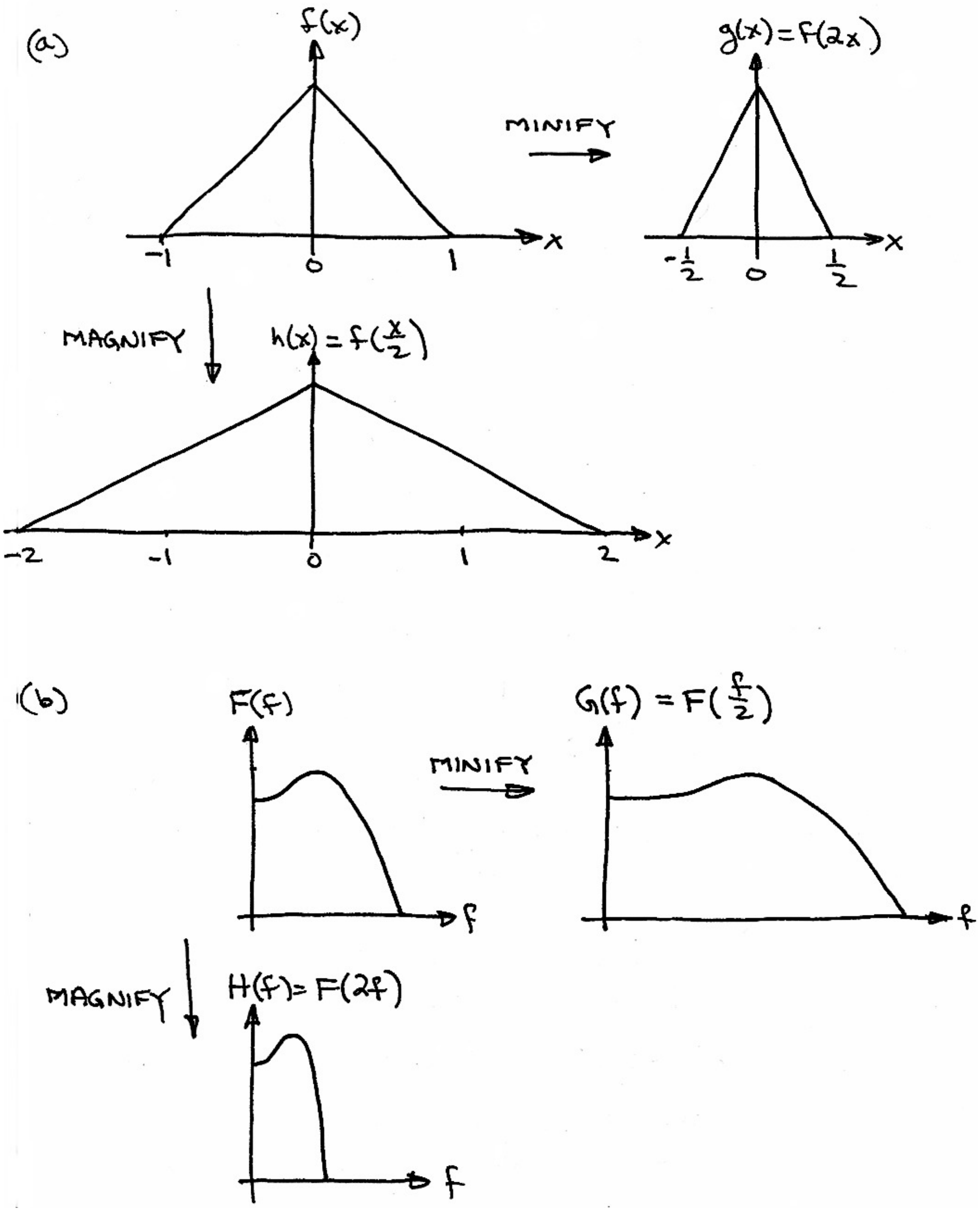

(a)
f(x)
MINIFY
g(x) = f(2x)
x
-1
0
1
$-\frac{1}{2}$
$\frac{1}{2}$
MAGNIFY
h(x) = f(x/2)
-2
2
(b)
F(f)
MINIFY
G(f) = F(f/2)
f
MAGNIFY
H(f) = F(2f)

Fig. 3. Magnification and minification in space and frequency.

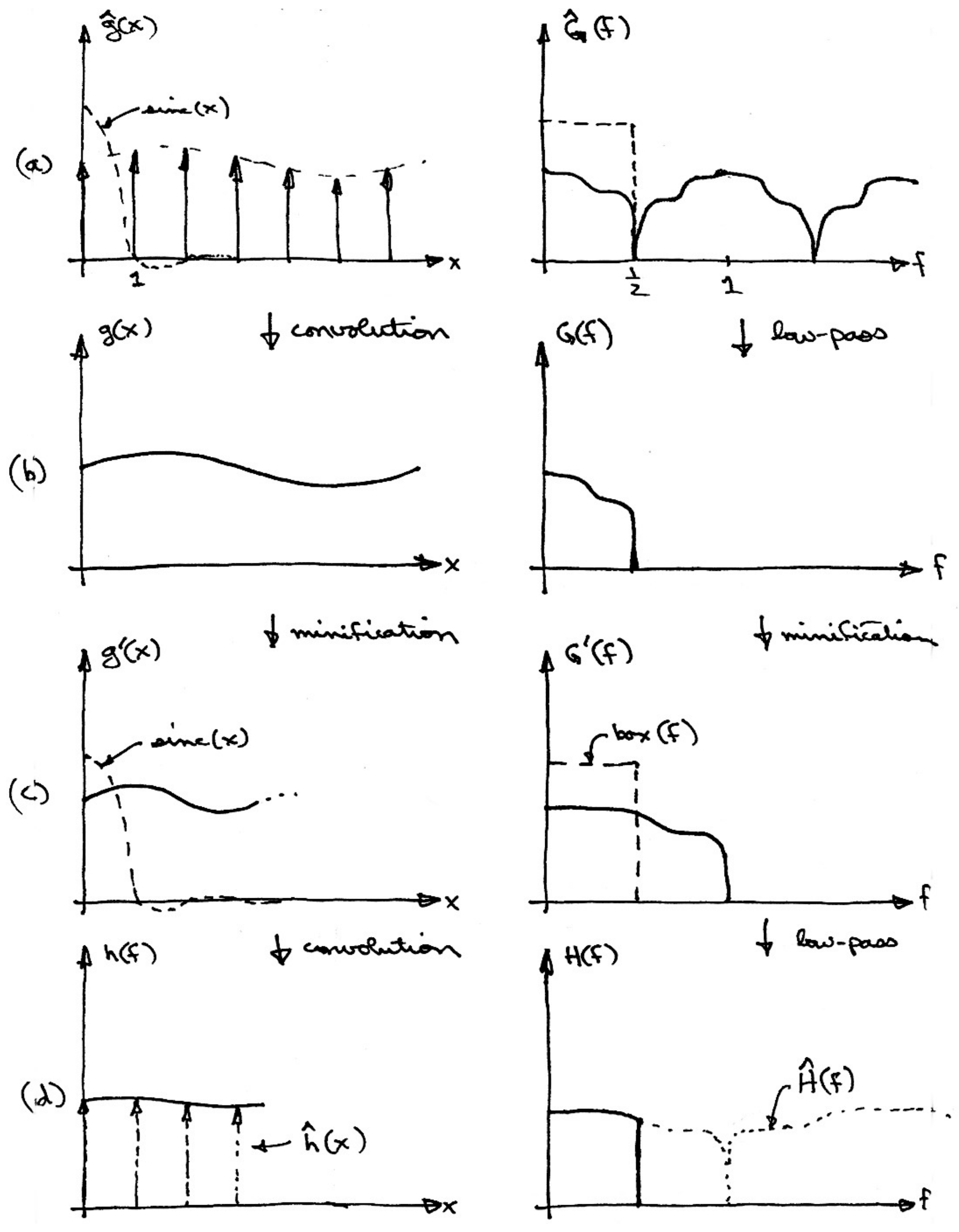


Fig. 4. The process of minification.

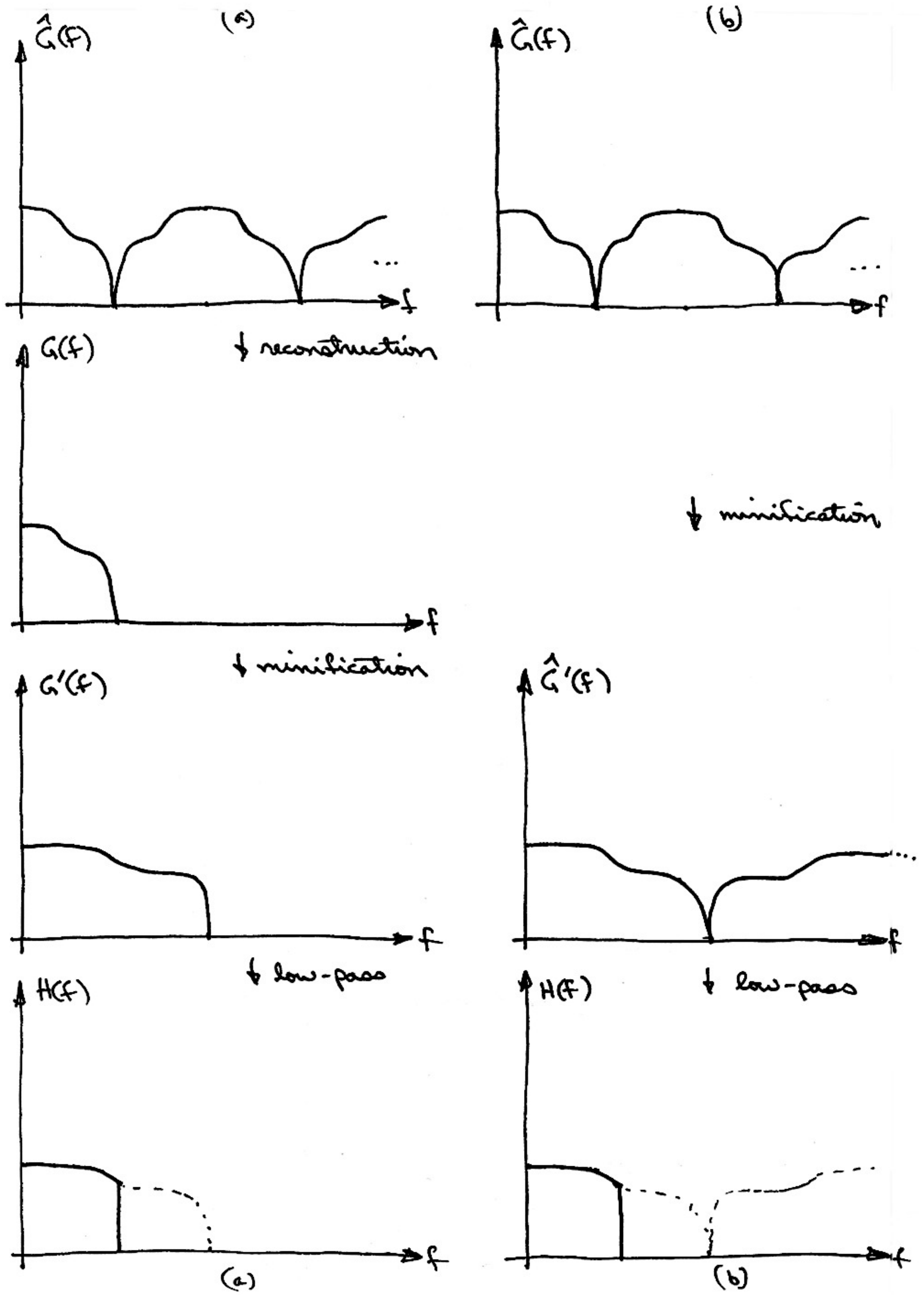
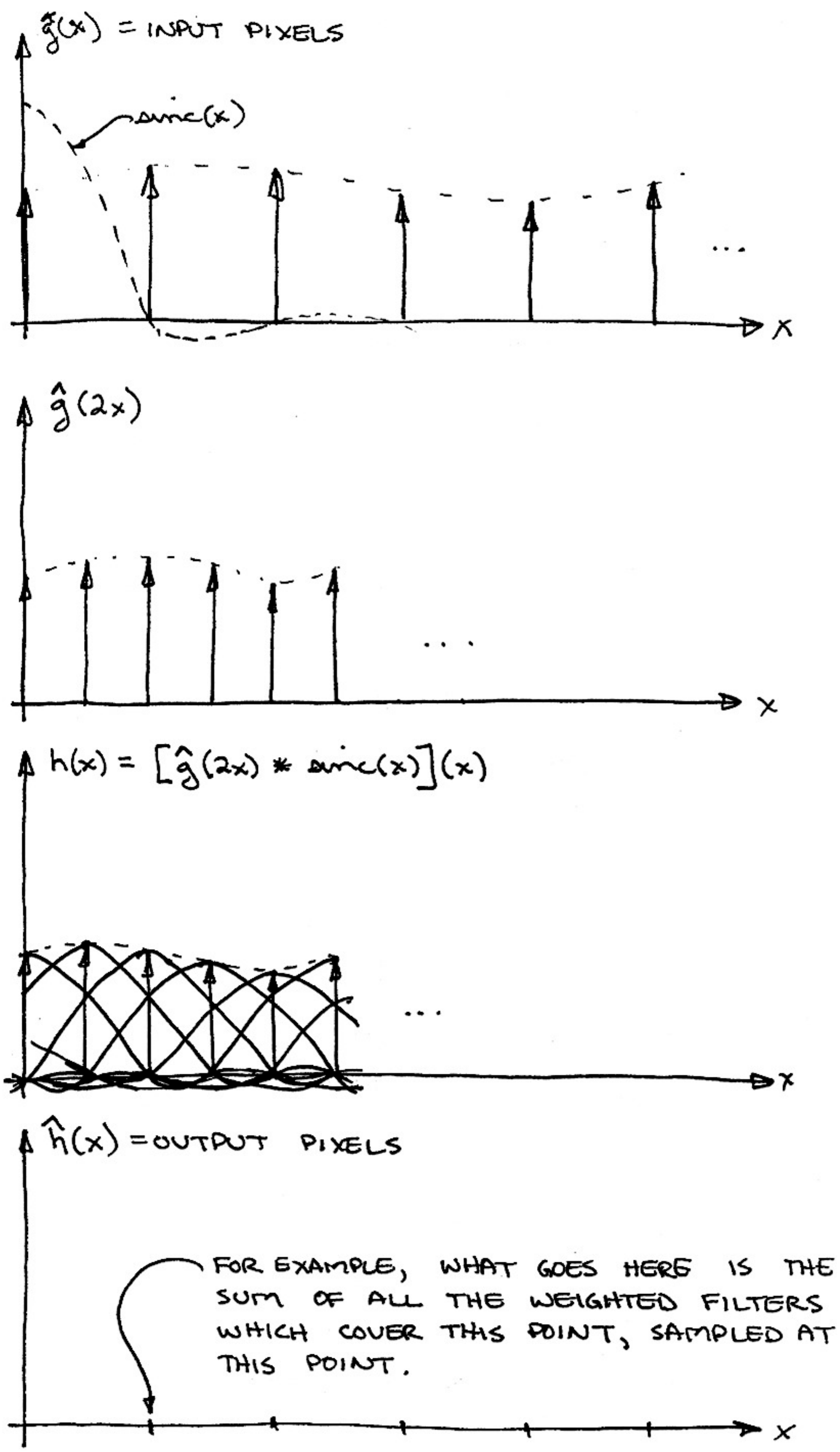

(a)
(b)
$\hat{G}(f)$
$\hat{G}(f)$
f
f
$G(f)$
reconstruction
f
minification
minification
$G'(f)$
$\hat{G}'(f)$
f
f
low-pass
low-pass
$H(f)$
$H(f)$
f
f
(a)
(b)

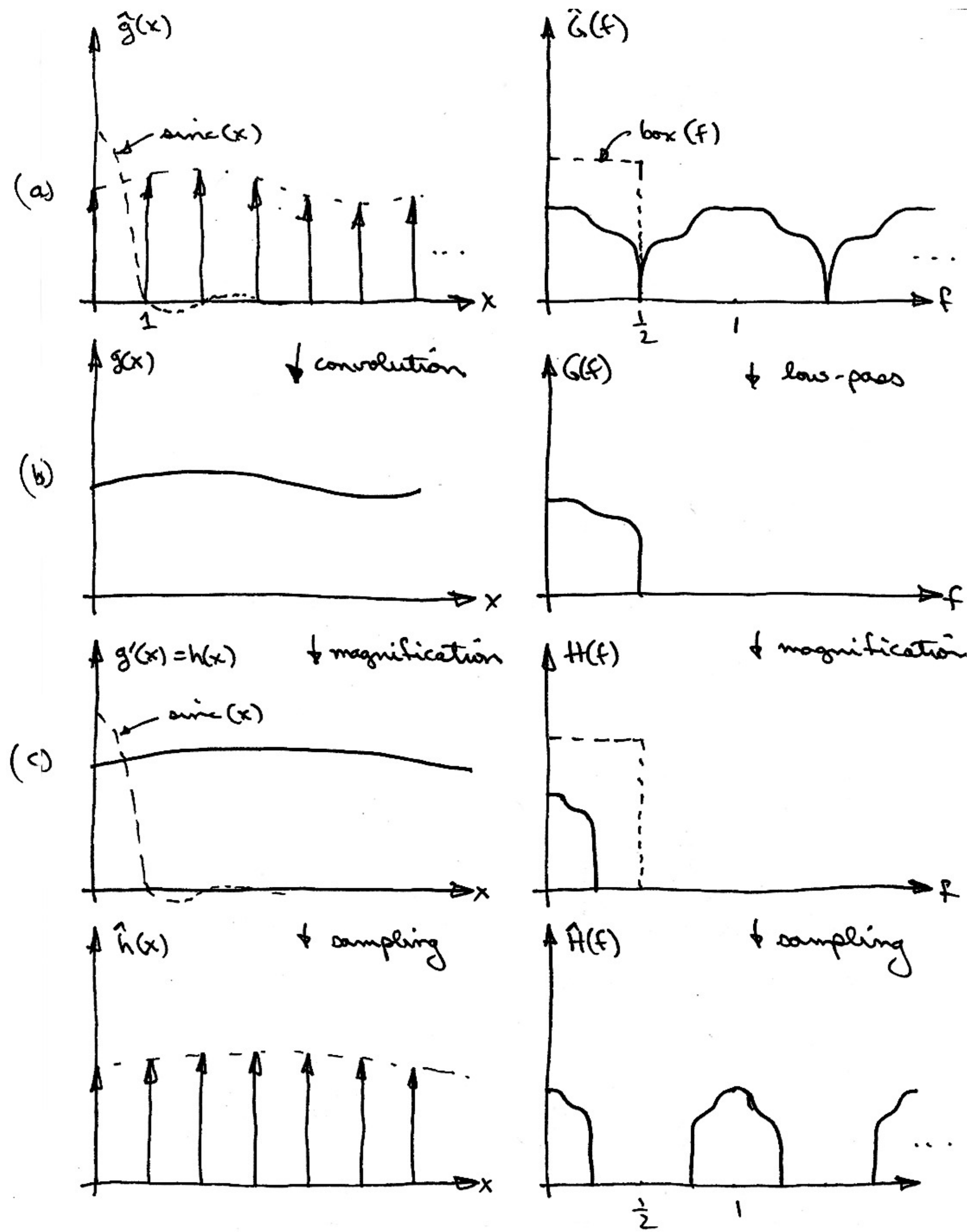
Fig. 5. The hard and easy ways to minify.



$\hat{g}(x)$ = INPUT PIXELS
sinc(x)
...
x
$\hat{g}(2x)$
...
x
$h(x) = [\hat{g}(2x) * \mathrm{sinc}(x)](x)$
...
x
$\hat{h}(x)$ = OUTPUT PIXELS
FOR EXAMPLE, WHAT GOES HERE IS THE SUM OF ALL THE WEIGHTED FILTERS WHICH COVER THIS POINT, SAMPLED AT THIS POINT.
x

Fig. 6. Details of the minification process.

Fig. 7. The process of magnification.



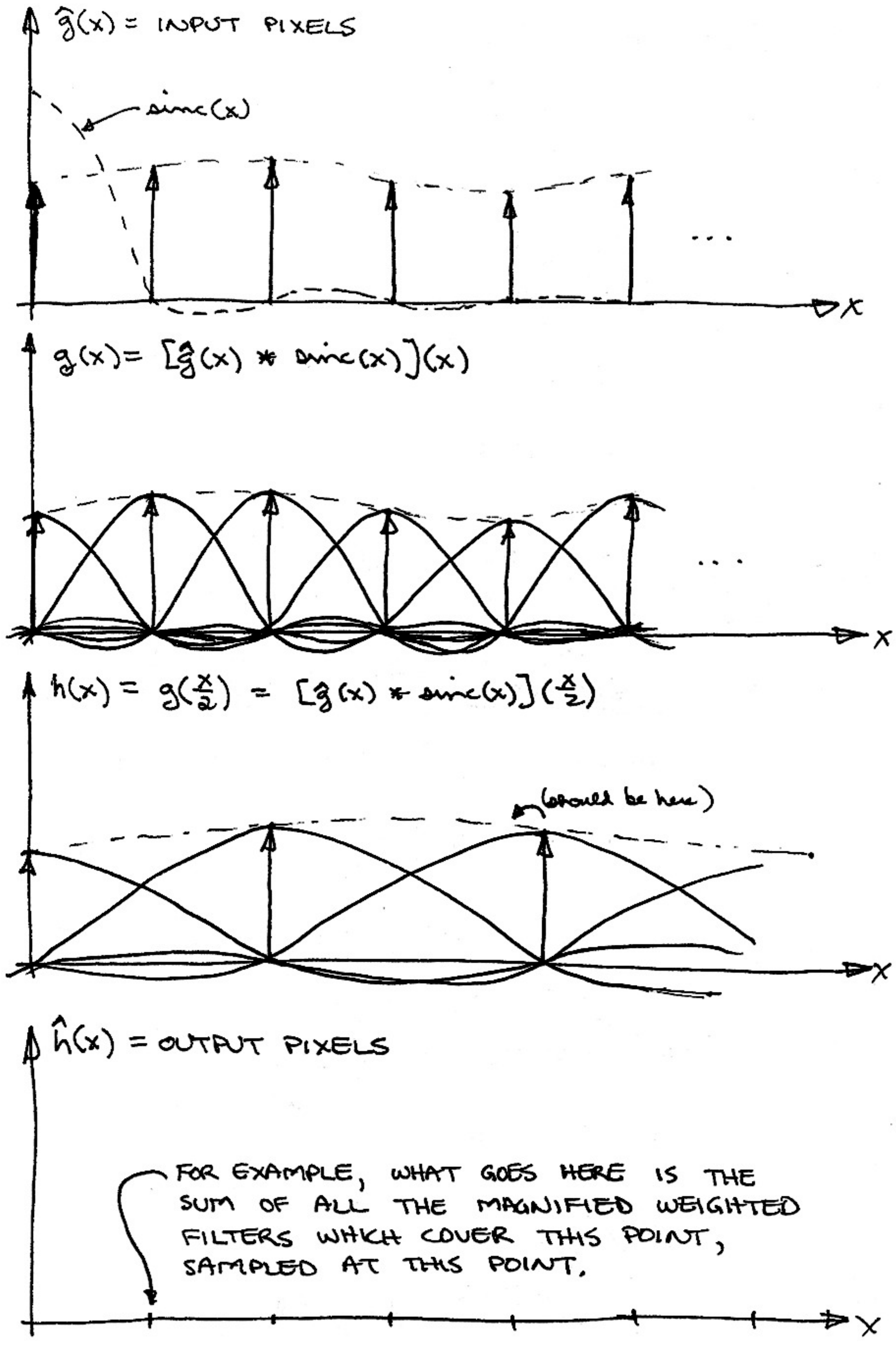

$\hat{g}(x)$ = INPUT PIXELS
sinc(x)
. . .
x
$g(x) = [\hat{g}(x) * \mathrm{sinc}(x)](x)$
. . .
x
$h(x) = g(\frac{x}{2}) = [\hat{g}(x) * \mathrm{sinc}(x)](\frac{x}{2})$
(should be here)
x
$\hat{h}(x)$ = OUTPUT PIXELS
FOR EXAMPLE, WHAT GOES HERE IS THE SUM OF ALL THE MAGNIFIED WEIGHTED FILTERS WHICH COVER THIS POINT, SAMPLED AT THIS POINT.
x

Fig. 8. Details of the magnification process.

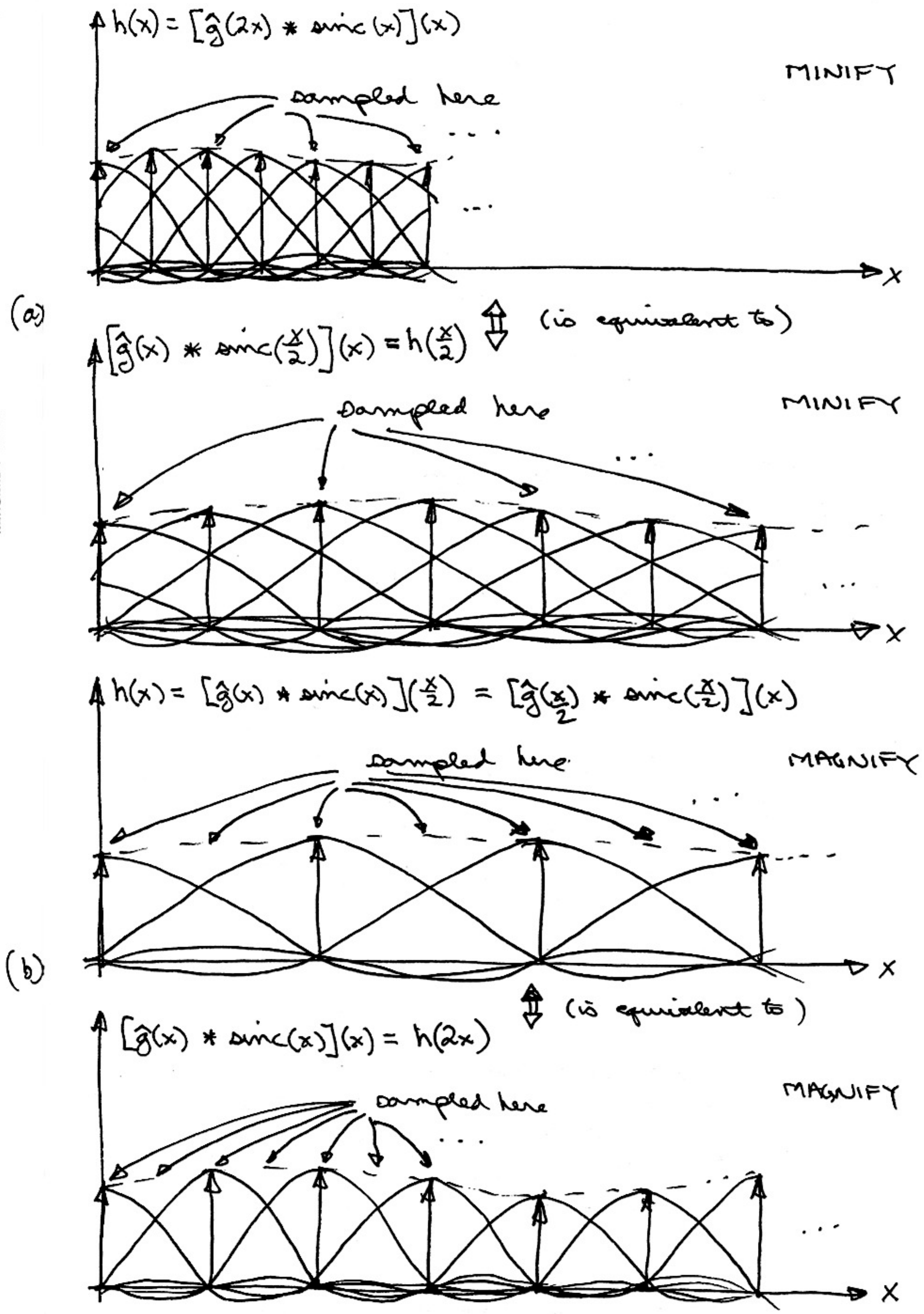
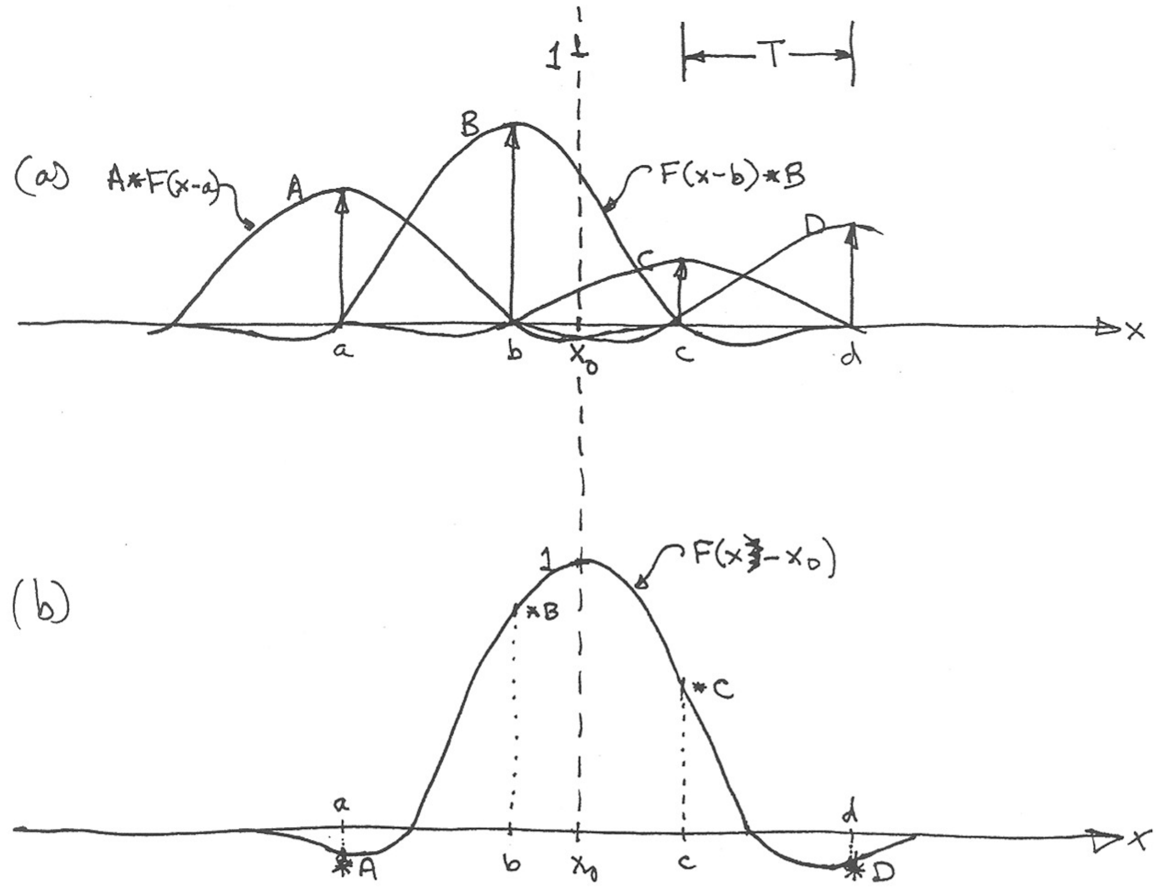

$h(x) = [\hat{g}(2x) * \mathrm{sinc}(x)](x)$
MINIFY
sampled here
(a)
(is equivalent to)
$[\hat{g}(x) * \mathrm{sinc}(\frac{x}{2})](x) = h(\frac{x}{2})$
MINIFY
sampled here
$h(x) = [\hat{g}(x) * \mathrm{sinc}(x)](\frac{x}{2}) = [\hat{g}(\frac{x}{2}) * \mathrm{sinc}(\frac{x}{2})](x)$
sampled here
MAGNIFY
(b)
(is equivalent to)
$[\hat{g}(x) * \mathrm{sinc}(x)](x) = h(2x)$
sampled here
MAGNIFY
x

Fig. 9. Theorem 1 applied to magnify/minify.

Fig. 10. Theorem 2 illustrated.



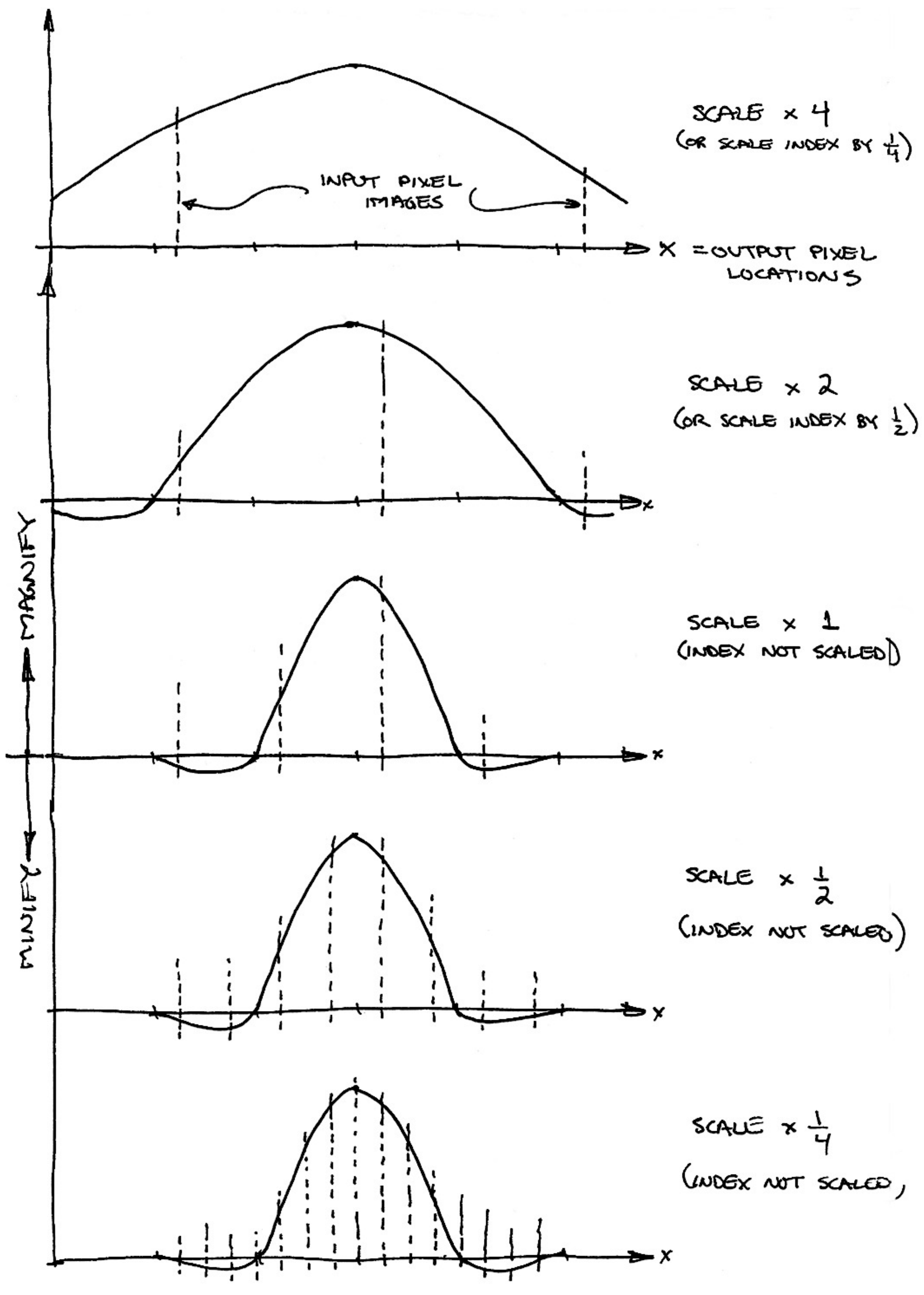

SCALE x 4
(OR SCALE INDEX BY 1/4)
INPUT PIXEL IMAGES
X = OUTPUT PIXEL LOCATIONS
SCALE x 2
(OR SCALE INDEX BY 1/2)
x
SCALE x 1
(INDEX NOT SCALED)
x
SCALE x 1/2
(INDEX NOT SCALED)
x
SCALE x 1/4
(INDEX NOT SCALED,
x
MAGNIFY
MINIFY

Fig. 11. Filter changes with scale factor.