Chapter 7: Image Decimation, Interpolation and Sampling Rate Conversion

1-D Decimation and Interpolation Theory

Decimation: Decimation by a factor of M can be modeled in two steps:

1. Multiply a sample array by a sparse impulse train, called the sub-sampling function.
2. Remove zero samples.

$$g(n) = f(n).\sum_{i=-\infty}^{\infty}\delta(n-iM) = g(Mn)$$

Decimation will result in aliasing if the resulting sampling rate is un the Nyquist rate. This can be shown mathematically in the Fourier domain. The Fourier series of the periodic sub-sampling function can be written by:

$$\sum_{i=-\infty}^{\infty}\delta(n-iM) = \frac{1}{M}\sum_{k=0}^{M-1}e^{j\frac{2\pi}{M}k.n}$$

Write the Fourier series expansion of the sampling function *g(n)*:

$$g(n) = f(n).\sum_{i=-\infty}^{\infty}\delta(n-iM) = g(Mn)$$

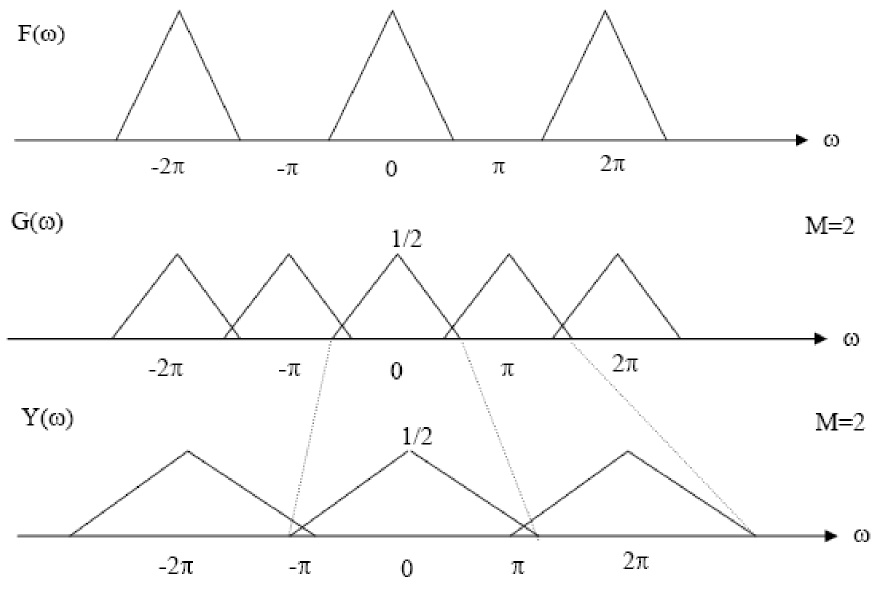
$$G(w) = \sum_{n=-\infty}^{\infty}g(n).e^{-jwn} = \frac{1}{M}\sum_{n=-\infty}^{\infty}f(n).\left\{\sum_{k=0}^{M-1}e^{j\frac{2\pi}{M}k.n}\right\}.e^{-jwn}$$

$$= \frac{1}{M}\sum_{k=0}^{M-1}\left\{\sum_{n=-\infty}^{\infty}f(n).e^{-j\left(w-\frac{2\pi}{M}k\right)n}\right\} = \frac{1}{M}\sum_{k=0}^{M-1}F(w-\frac{2\pi}{M}k)$$

We now write the Fourier transform of the decimated result *y(n)* by using a simple change of variables $\eta = Mn$

$$Y(w) = F\{g(Mn)\} = \sum_{n=-\infty}^{\infty}g(n).e^{-j\frac{w}{M}n} = G(\frac{w}{M}) = \frac{1}{M}\sum_{k=0}^{M-1}F(\frac{w-2\pi k}{M}) \quad for\ -\pi \le w < \pi$$

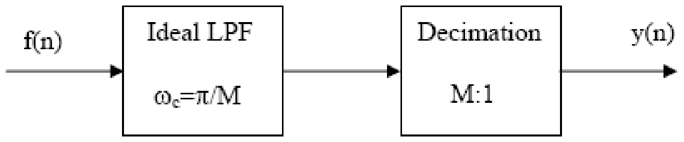
It is the usual scaling property of the Fourier transform.

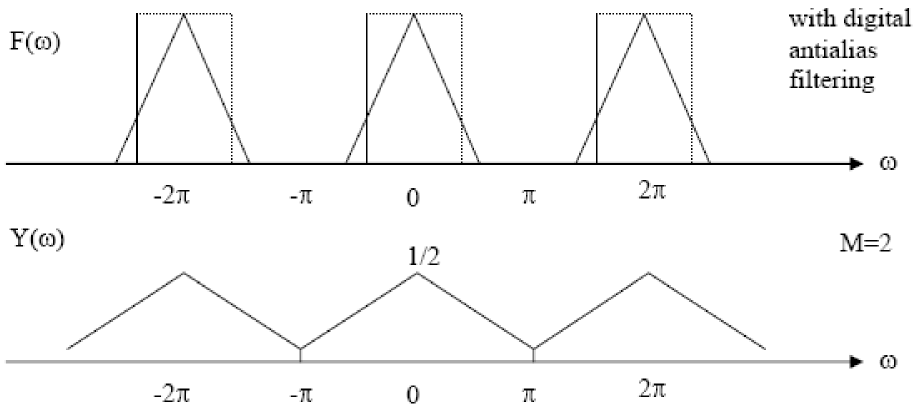


Clearly, if the signal was over-sampled by at least a factor of *M*, i.e: $F(w) = 0 \quad for \quad \pi / M \leq |w| < \pi$ then there will be no aliasing after decimation. However, if this condition is NOT met then a digital low-pass filter with a frequency response:

$$H(w) = \begin{cases} 1 & if \quad 0 \leq |w| < \dfrac{\pi}{M} \\ 0 & if \quad \dfrac{\pi}{M} \leq w < \pi \end{cases}$$

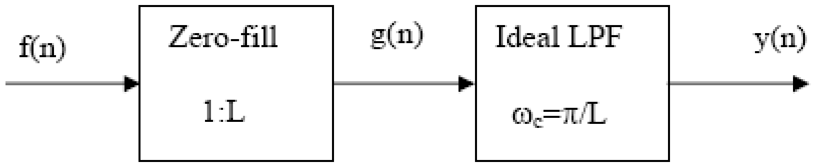
should be applied to the input *f(n)* prior to decimation to avoid aliasing, which is called a digital anti-alias filter, as illustrated below.





Interpolation: Interpolation by a factor of L can be similarly realized in two steps:

1. Fill-in L-1 zeros in between every sample of the input signal to obtain a higher rate signal.
2. Followed by proper low-pass filtering as shown below.



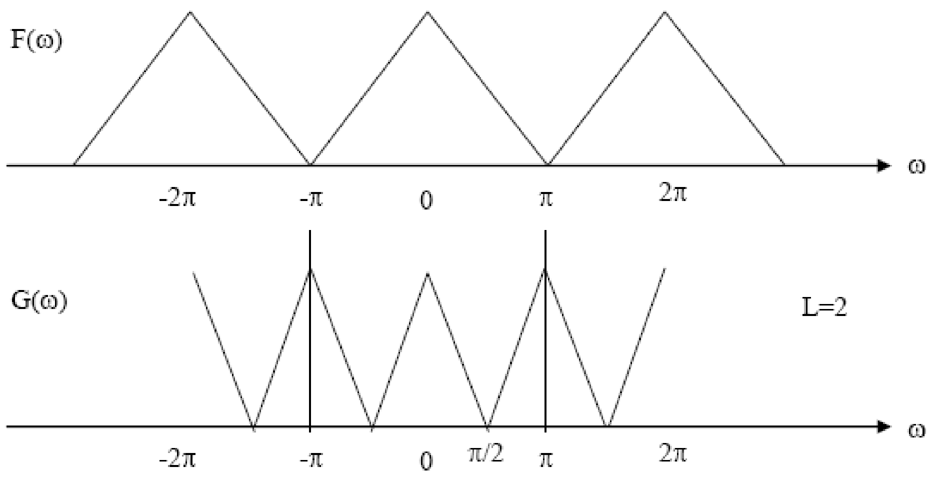
Zero-filling operation can be expressed by:

$$g(n) = \begin{cases} f(\frac{n}{L}) & if \;\; n = 0, \pm L, \pm 2L, \cdots \\ 0 & Otherwise \end{cases}$$

Opposite to the case of decimation the effect of zero-fill in the frequency domain is a compression of the frequency axis as seen by:

$$G(w) = \sum_{n=-\infty}^{\infty} f(n/L).e^{-jwn} = \sum_{n'=-\infty}^{\infty} f(n').e^{-jwL.n'} = F(wL)$$

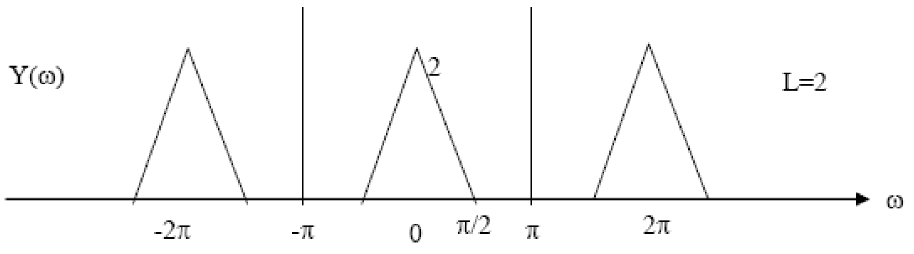
where we employ the change of variables *n'=n/L*. This is iluustrated below.



Corresponding ideal low-pass filter to remove unwanted copies in the interval $-\pi < \omega \leq \pi$ will be:

$$H_{\text{int}}(w) = \begin{cases} L & \text{if } w < \pi / L \\ 0 & \text{Otherwise} \end{cases}$$

This results in an interpolated signal *y(n).*



Sampling Rate Change by a Rational Factor: Sampling rate change by a rational factor of *L/M* corresponds to:

1. Interpolation by an integer factor of L followed by
2. Decimation by an integer factor of M.

Therefore, we can cascade the two systems we learned above as shown below to realize sampling rate change by a rational factor.

$$\begin{tikzcd} f(n) \arrow[r] & \boxed{\begin{matrix}\text{Zero-fill}\\ 1:L\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{Ideal LPF}\\ \omega_c=\pi/L\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{Ideal LPF}\\ \omega_c=\pi/M\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{Decimation}\\ M:1\end{matrix}} \arrow[r] & y(n) \end{tikzcd}$$

Clearly, the back-to-back low-pass filters can be combined to simplify the system:

$$\begin{tikzcd} f(n) \arrow[r] & \boxed{\begin{matrix}\text{Zero-fill}\\ 1:L\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{Ideal LPF}\\ \omega_c=\min\{\pi/M, \pi/L\}\end{matrix}} \arrow[r] & \boxed{\begin{matrix}\text{Decimation}\\ M:1\end{matrix}} \arrow[r] & y(n) \end{tikzcd}$$

Efficient and yet eloquent implementation of this system is usually done by using polyphase filters in the DSP community.

Practical Image Decimation and Interpolation

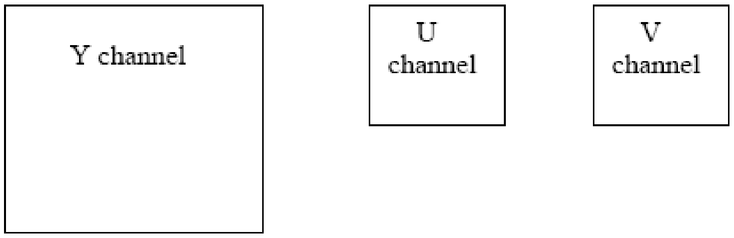
Both decimation and interpolation require ideal low-pass filtering. The impulse response of an ideal low-pass filter being a *Sinc(x)* function has infinitely many periodic zero-crossings, clearly, interpolation by a *Sinc* function in the frequency-domain is unrealizable. Therefore, a realizable approximation must replace it.

Decimation by Averaging: Generally, simple averaging or Gaussian weighting with the parameter σ is employed in the decimation process.

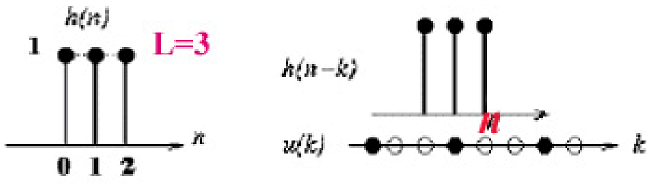
$$h(n_1,n_2) = K.\exp(-\frac{n_1^2+n_2^2}{2\sigma^2})$$

where K is a normalization factor such that summation of all weights equals 1.

Example: 4:2:0 Decimation of color information (chrominance) used in many video compression standards.



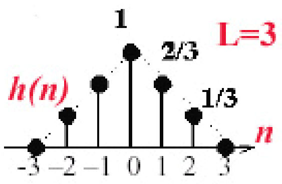
Zero-Order Hold Interpolation:

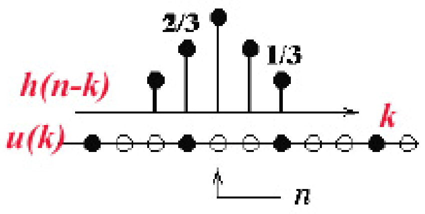


This is the simplest form of interpolation also known as pixel replication method. The impulse response of the zero-order hold filter is a rectangular function (length L), as shown below, which implies that its frequency response is given by a Sinc(x) function with the first zero crossing at $w = 2\pi / L$.

$$h(n) = \begin{cases} 1 & if \ \ 0 \le n \le L-1 \\ 0 & Otherwise \end{cases} \qquad H(w) = \sum_{n=0}^{L-1} e^{-jwn} = e^{-jw(L-1)/2} . \frac{Sin(wL/2)}{Sin(w/2)}$$

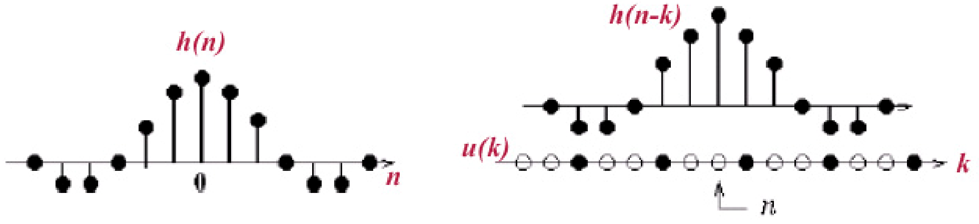
Linear Interpolation (First-order Hold): Linear interpolation is probably the most popular choice because of simplicity of its implementation. The impulse response of the linear interpolation filter is given by the triangle function, which implies its frequency response is given by (Sinc)^2 function.





$$h(n) = \begin{cases} \frac{L-|n|}{L} & if \ \ 0 \le |n| \le L-1 \\ 0 & Otherwise \end{cases}$$

Cubic Convolution Interpolation: Cubic convolution interpolation kernel approximates the impulse response of the ideal lowpass filter (Sinc function) by three cubic polynomial pieces with a significantly better frequency response.



Cubic convolution interpolation kernel is used for converting the samples into a continuous signal. Then the desired interpolation ratio is obtained by re-sampling the continuous signal. Continuous cubic convolution interpolation kernel in the interval (-2, 2) is given by:

$$h(t) = \begin{cases} a_3|t|^3 + a_2|t|^2 + a_1|t| + a_0 & 0 \le t < 1 \\ b_3|t|^3 + b_2|t|^2 + b_1|t| + b_0 & 1 \le |t|\, 2 \\ 0 & |t| \ge 2 \end{cases}$$

which implies that we need use four existing samples (two on each side) to perform the interpolation. The following regularity constraints are imposed on the kernel:

$$h(0) = a_0 = 1 \qquad h(1^-) = a_3 + a_2 + a_1 + a_0 = 0 \quad \text{and} \quad h(1^+) = b_3 + b_2 + b_1 + b_0 = 0$$

$$h(2) = 8b_3 + 4b_2 + 2b_1 + b_0 = 0 \qquad \frac{d}{dt}h(0^-) = -a_1 = \frac{d}{dt}h(0^+) = a_1 \Rightarrow a_1 = 0$$

$$\frac{d}{dt}h(1^-) = 3a_3 + 2a_2 + a_1 = \frac{d}{dt}h(1^+) = 3b_3 + 2b_2 + b_1$$

$$\frac{d}{dt}h(2) = 12b_3 + 4b_2 + b_1 = 0$$

As a result, we have 7 equations in 8 unknowns. Thus, we let $b_3 = p$, a free parameter. Solving these 7 equations in terms of *p* gives the following interpolation kernel:

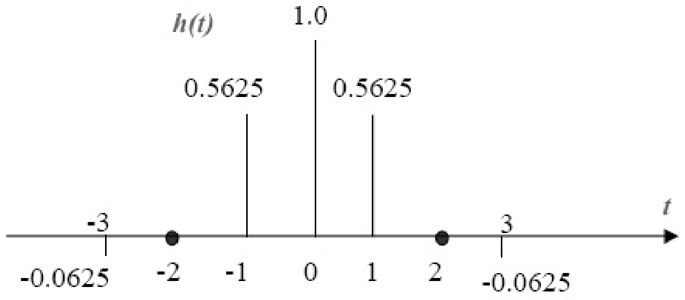
$$h(t) = \begin{cases} (p+2)|t|^3 - (p+3)|t|^2 + 1 & 0 \le t < 1 \\ p|t|^3 - 5p|t|^2 + 8p|t| - 4p & 1 \le |t|\, 2 \\ 0 & |t| \ge 2 \end{cases}$$

If we require the interval $0 \le t < 1$ to be concave upward with $\frac{d^2}{dt^2}h(0) < 0$ and $1 \le t < 2$ to be concave downward with $\frac{d^2}{dt^2}h(0) > 0$ and we restrict the range of *p* to *-3 < p <0*. The most common choices are *p=-0.5* or *p=-1*. The impulse response of the cubic convolution interpolator for any value of *L* can be obtained by sampling *h(t)* with *4L+1* samples in the interval $-2 \le t < 2$..
Since *h(-2)=h(2)=0*, this implies *4L-1* samples excluding end points. Finally, we can assume that Two-D interpolation kernel is separable, $h(n_1,n_2) = h_1(n_1)\, h_2(n_2)$.

Example: Cubic convolution interpolation kernel for $p = -0.5$ *and* $L = 2$ is found by substituting first $p = -0.5$ into above kernel equation:

$$h(t)=\begin{cases}\frac{3}{2}|t|^3-\frac{5}{2}|t|^2+1 & 0\le t<1\\ -\frac{1}{2}|t|^3+\frac{5}{2}|t|^2-4|t|+2 & 1\le|t|2\\ 0 & |t|\ge 2\end{cases}$$

The length of the impulse response for *L=2* is 7 (excluding the end points) and sampling points are *t=-1.5, -1, -0.5, 0, 0.5, 1, 1.5*. The impulse response is given by:



Wiener Interpolation Filtering: An optimal interpolation filter design strategy is to minimize the mean square interpolation error using the correlation function of the image (to be interpolated). For a filter with symmetric impulse response, interpolation by a factor of *L=2* can be expressed as:

$$y(2n)=f(2n)$$

$$y(2n+1)=\sum_{l=1}^{L_W}h(l).[f(2(n-l+1))+f(2(n+l))]$$

In the usual optimization procedure based on mean-square error minimization, we differentiate the expected value of the error function with respect to *h(k)* and set it to zero:

$$E\{[f(2n+1)-y(2n+1)]^2=E\{[f(2n+1)-\sum_{l=1}^{L_W}h(l).[f(2(n-l+1))+f(2(n+l))]]^2\}$$

and solve for unknowns:
which gives Lw equations Lw in unknowns as

$$\frac{\partial}{\partial h(k)}E\{[f(2n+1)-\sum_{l=1}^{L_W}h(l).[f(2(n-l+1))+f(2(n+l))]]^2\}=0 \quad for\ k=1,2,\cdots,L_W$$

$$2.E\{[f(2n+1)-\sum_{l=1}^{L_W}h(l).[f(2(n-l+1))+f(2(n+l))]].[f(2(n-k+1))+f(2(n+k))]\}=0$$

$$for\ k=1,2,\cdots,L_W$$

The inner products are nothing but correlation function $R_f(l)$ of the image *f(n)* and this system of linear equations. We now need to solve L_W linear equations in L_W unknowns:

$$\sum_{l=1}^{L_W}h(l).[R_f(2(k-l))+R_f(2(k+l-1))]=R_f)2k-1) \quad for\ k=1,2,\cdots,L_W$$

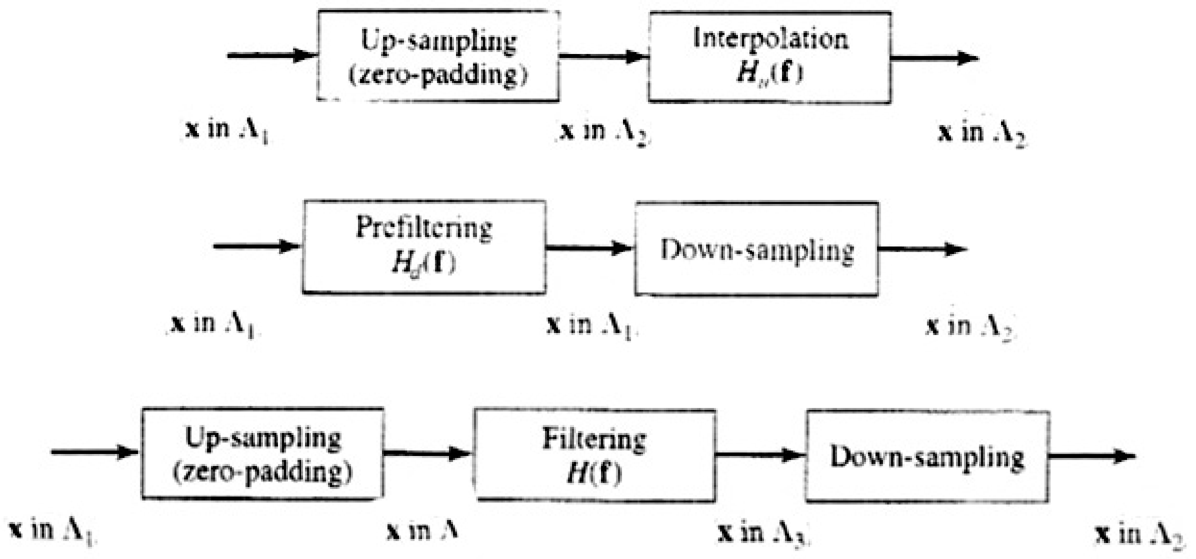
subject to the constraint: $\sum_{l=1}^{L_W} 2h(l) = 1$. These equations are very frequently seen in communications, DSP, and optimization theory, there are tedious techniques involving matrix inversions or Levinson Recursion techniques (Durbin, Burg, La Roux-Geugen and other algorithms) as used in speech prediction.

Decimation and Interpolation on Lattices: Let us define sum (more appropriately "union" and intersection of two lattices by:

$$\Lambda_1 + \Lambda_2 = \{x_1 + x_2;\, x_1 \in \Lambda_1 \textit{ and } \; x_2 \in \Lambda_2\} \quad \Lambda_1 \cup \Lambda_2 = \{x;\, x \in \Lambda_1 \textit{ and } \; x \in \Lambda_2\}$$

Conversion from a lattice Λ_1 to another lattice Λ_2 can be modeled by up-conversion from Λ_1 to the sum of $\Lambda_1 + \Lambda_2$ followed by ideal LPF on $\Lambda_1 + \Lambda_2$, followed by down-conversion to Λ_2.

General block diagram for rate conversion, which is the standard process for NTSC-to-PAL and the inverse conversion, is shown below.



Sampling rate conversion based on lattices: Up-conversion (top); down-conversion (middle), and arbitrary rate conversion (bottom).

Up-conversion: Since $\Lambda_1 \subset \Lambda_2$, we can transfer all samples in Λ_1 into Λ_2 and fill the missing places with zeros:

$$g_{S,3}(x) = \begin{cases} f_{S,1}(x) & \textit{if} \quad x \in \Lambda_1 \\ 0 & \textit{if} \quad x \in \Lambda_2 \setminus \Lambda_1 \end{cases}$$

where $g_{S,3}(x)$ stands for the samples at the output of up-sampler and $\Lambda_2 \setminus \Lambda_1$ represents set of point in Λ_2 but not in Λ_1. To fill zero-padded samples, we need to apply an interpolation filter. Let us sample the input signal (continuous version) directly over Λ_2 with a generating matrix V_2, and the sampling density: $d(\Lambda_2) = 1/|\det(\Lambda_2)|$ then the FT of the sampled signal would be:

$$Y_{S,2}(f) = d(\Lambda_2).\sum_m X(f - U_2 m) \quad \text{with the property: } U_2 = (V_2^T)^{-1}$$

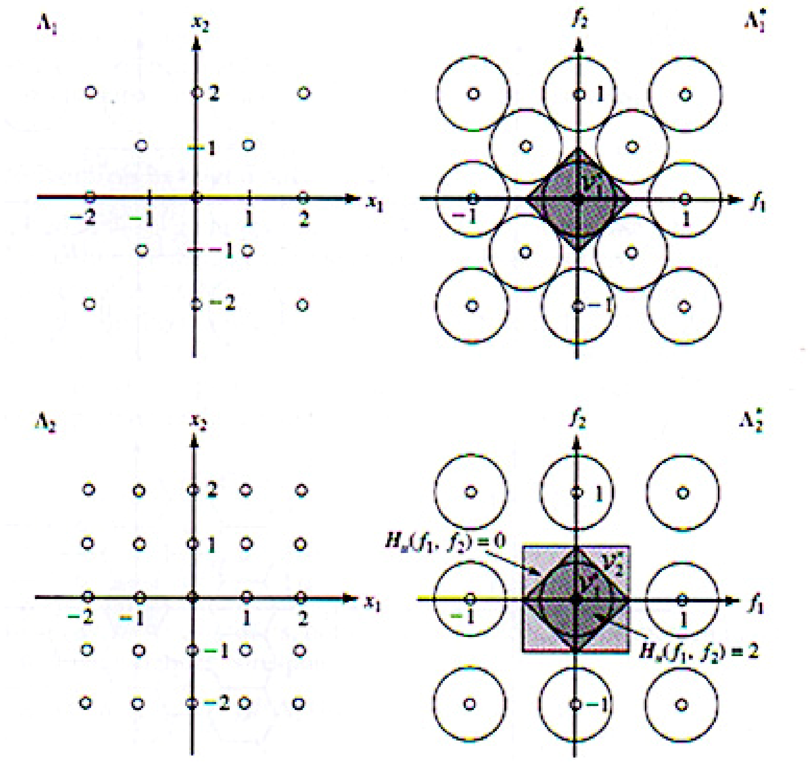
On the other hand, the FT of the signal sampled over Λ_1 with a generating matrix V_1 is given by:

$$X_{S,1}(f) = d(\Lambda_1).\sum_m X(f - U_1 m) \quad \text{with the property: } U_1 = (V_1^T)^{-1}$$

Assuming that Λ_1 satisfies the alias-free condition then there will be no overlapping between the alias components in Λ^*_1. Under these conditions, the frequency response of the "ideal" interpolation filter would be:

$$H_{up}(f) = \begin{cases} d(\Lambda_2)/d(\Lambda_2) & f \in V_1^* \\ 0 & f \in V_2^* \setminus V_1^* \end{cases}$$

Example: Consider the up-conversion in the following figure. Suppose that the original signal has a circular support, (it could also be a diamond as well). It is clear that $\Lambda_1 \subset \Lambda_2$



Matrices in this case are: $V_1 = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$ *and* $V_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Their reciprocal lattices would be: $U_1 = \begin{bmatrix} 1/2 & 0 \\ -1/2 & 1 \end{bmatrix}$ *and* $U_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Reciprocal lattices and their Voronoi cells are also shown in the figure. It is also clear that $V_2^* \supset V_1^*$ and the ideal filter in this case would be:

$$H_{up}(f) = \begin{cases} d(\Lambda_2)/d(\Lambda_1) = 2 & f \in V_1^* \\ 0 & f \in V_2^* \setminus V_1^* \end{cases}$$

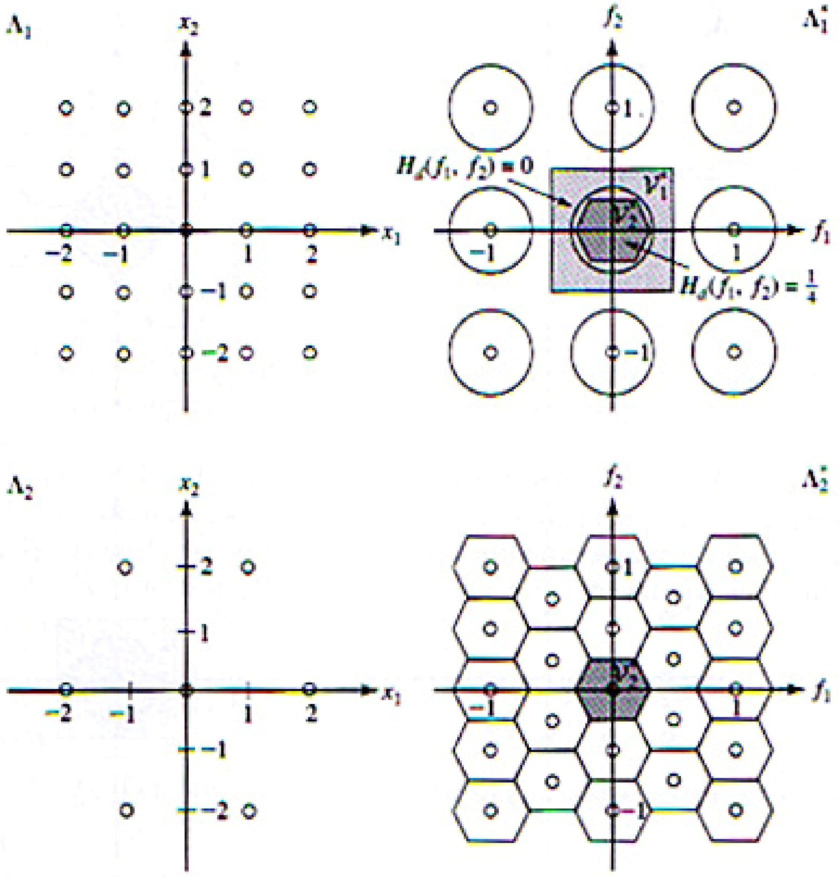
as illustrated on lattice V_2^*. This filter can be used to eliminate alias components of V_1^* that should not appear in V_2^*.

Down-conversion: In this case, $\Lambda_1 \supset \Lambda_2$ and we can obtain $Y_{S,2}(X)$ by retaining all samples in Λ_1 that are also in Λ_2 and discard all others:

$$g_{S,3}(x) = f_{S,1}(x) \quad for\ x \in \Lambda_2$$

This signal as obtained would be exactly the same as the one resulting from sampling the original continuous signal over Λ_2, if there was one or obtained through D/A conversion. Suppose that the support region of the original continuous signal is equal or smaller than V_1^*, so that there is nop aliasing in $f_{S,1}(x)$. But there will be aliasing in $Y_{S,2}(f)$ as V_2^* is smaller. To avoid aliasing then we need to pre-filter the input with a filter:

$$H_{down}(f) = \begin{cases} d(\Lambda_2)/d(\Lambda_1) & f \in V_2^* \\ 0 & f \in V_1^* \setminus V_2^* \end{cases}$$



Example: Consider the down-conversion in the above figure.

$$V_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad and \quad V_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$U_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad and \quad U_2 = \begin{bmatrix} 1/2 & 0 \\ -1/4 & 1/2 \end{bmatrix}$$

The ideal pre-filter is given by: $H_{down}(f) = \begin{cases} 1/4 & f \in V_2^* \\ 0 & f \in V_1^* \setminus V_2^* \end{cases}$

Without pre-filtering, the original circular spectrum would have caused alasing in V_2^*. By applying $H_{down}(f)$ in V_1^*, only the portion of the spectrum contained in V_2^* is retained.

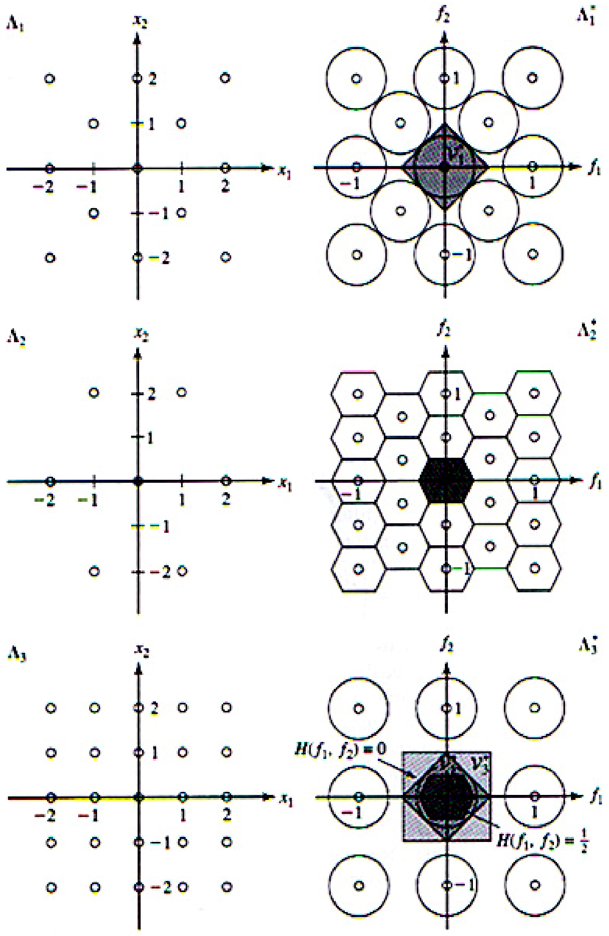
Arbitrary rate-conversion: When Λ_1 and Λ_2 are not subsets of each other, as in the case of NTSC to PAL conversion rate, we need to introduce another lattice Λ_3, smallest one as possible, so that the other two are common divisors of this new large lattice. Then the process is the concatenation of the up-conversion and down-conversion cases discussed above. The interpolation filter and the pre-filter in this case would be:

$$H_{up}(f) = \begin{cases} d(\Lambda_3)/d(\Lambda_1) & f \in V_1^* \\ 0 & f \in V_3^* \setminus V_1^* \end{cases}$$

$$H_{down}(f) = \begin{cases} d(\Lambda_2)/d(\Lambda_3) & f \in V_2^* \\ 0 & f \in V_3^* \setminus V_2^* \end{cases}$$

It is common practice to merge these two filters into one described by:

$$H_{U-D}(f) = \begin{cases} d(\Lambda_2)/d(\Lambda_1) & f \in V_1^* \cap V_2^* \\ 0 & f \in V_3^* \setminus V_1^* \cap V_2^* \end{cases}$$



Example: Consider the conversion between two sampling lattices Λ_1 and Λ_2 shown above with generating matrices:

$$V_1 = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \quad and \quad V_2 = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$$

$$U_1 = \begin{bmatrix} 1/2 & 0 \\ -1/2 & 1 \end{bmatrix} \quad and \quad U_2 = \begin{bmatrix} 1/2 & 0 \\ -1/4 & 1/2 \end{bmatrix}$$

In this case Λ_1 and Λ_2 are not subsets of each other, so, we need to use a new lattice Λ_3 that is smallest lattice which contains both. In this case, Λ_3 can be determined from by the sum of Λ_1 and Λ_2 if $V_1^{-1}.V_2$ contains only rational numbers, which is the generalization of the one-D case for $r = rational\ number = \Lambda_1 / \Lambda_2 = p/q$. In our case, a careful observation of the figure shows that Λ_3 must be the square lattice sketched in the figure and the identity matrix will do as the generating matrix:

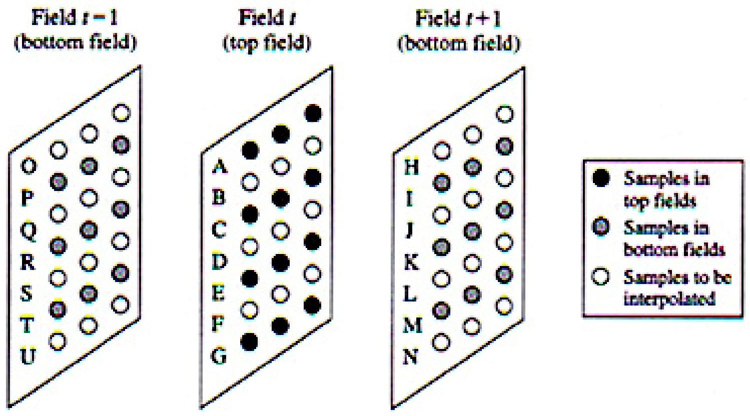
$$V_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad V^{-1}{}_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

From these three lattices, we can determine their Voronoi cells individually and the intermediate filter needs to be:

$$H_{up}(f) = \begin{cases} d(\Lambda_2)/d(\Lambda_1) = 1/2 & f \in V_2^* \\ 0 & f \in V_3^* \setminus V_2^* \end{cases}$$

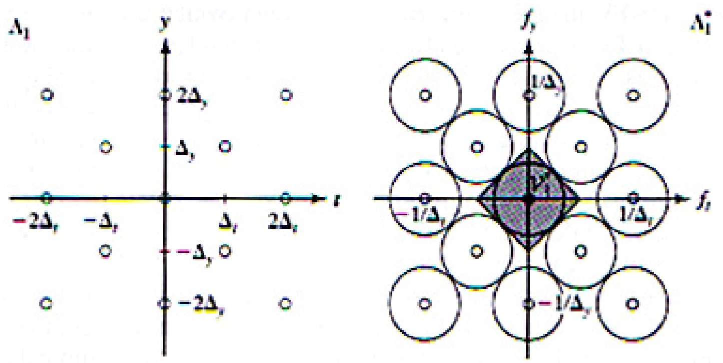
Sampling rate conversion of video: In many applications including the NTSC-to-PAL, and vice versa, translation, sampling rate conversion is needed. Recall that PAL signal on an NTSC system requires the conversion of an interlaced signal with a sampling rate of $F_{S,t} = 50 \frac{fields}{s}$ and $F_{S,y} = 625 \frac{lines}{frame}$ into another interlaced signal with $F_{S,t} = 60 \frac{fields}{s}$ and $F_{S,y} = 525 \frac{lines}{frame}$. Yet in many applications, one needs to convert an interlaced signal into a progressive raster, commonly known as "de-interlacing."

De-interlacing example: Consider the following scenario:

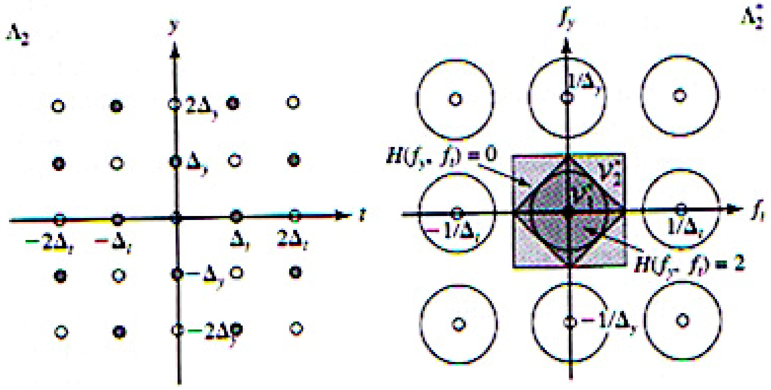


Sampling intervals: $\Delta t = 1/60\,s$ and $\Delta y = 1/525\, lines/\ frame$

Corresponding sampling lattice and its reciprocal is shown below. Note that the picture has been scaled to represent Δt and Δy with same length for graphical ease.



Sampling lattice and reciprocal corresponding to interlaced scan.



Sampling lattice and reciprocal after de-interlacing.

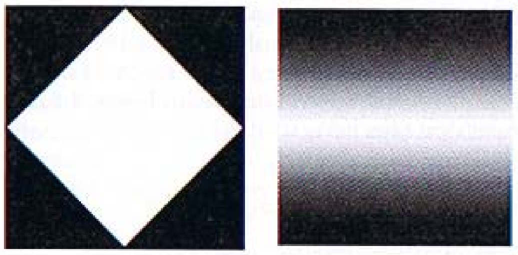
$$V_1 = \begin{bmatrix} 2.\Delta t & \Delta t \\ 0 & \Delta y \end{bmatrix} \quad \text{and} \quad U_1 = \begin{bmatrix} 1/2.\Delta t & 0 \\ -1/2.\Delta y & 1/\Delta y \end{bmatrix}$$

$$V_2 = \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta y \end{bmatrix} \quad \text{and} \quad U_2 = \begin{bmatrix} 1/\Delta t & 0 \\ 0 & 1/\Delta y \end{bmatrix}$$

In this case, since $\Lambda_1 \subset \Lambda_2$, the problem is an up-conversion problem and the interpolation filter is given by:

$$H_{up}(f) = \begin{cases} d(\Lambda_2)/d(\Lambda_1) = 2 & (F_y, F_t) \in V_1^* \\ 0 & (F_y, F_t) \in V_2^* \setminus V_1^* \end{cases}$$

The Voronoi cells of this filter is also shown above. The magnitude of this filter is shown below (left).



This is an ideal low-pass filter with a diamond shape pass-band. Since it cannot be decomposed into a product of a temporal filter and a vertical filter, it is not separable, which requires a two-D filter design. In practice, many simpler filters have been proposed for de-interlacing. The simplest is "**line**

averaging," which generates a missing line from the average of the line above and below. Using this for field *t*, we can use $D = (C + E)/2$ and the equivalent filter in Λ_2 will be:

$$h_{la}(y,t) = \begin{cases} 1 & (y,t) = (0,0) \\ 1/2 & (y,t) = (\Delta_y, 0), (-\Delta_y, 0) \\ 0 & Otherwise \end{cases}$$

and the frequency response of this filter is given by:

$$H_{la}(f_y, f_t) = 1 + \frac{1}{2} e^{j2\pi\Delta_y f_y} + \frac{1}{2} e^{-j2\pi\Delta_y f_y} = 1 + Cos(2\pi\Delta_y f_y)$$

which is shown above (right). Higher-order averaging (more lines) can be used to improve this vertical interpolation.

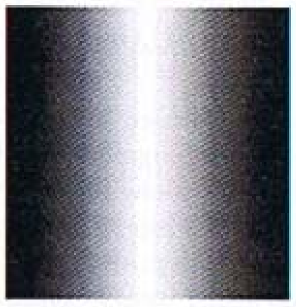
Alternately, we can use temporal interpolation as well. Since there is line in the next field for every missing line in the current field, we can copy this corresponding line: $D = K;\ J = C, etc.$ This is called "**field merging.**" Corresponding and the equivalent filter and its frequency response for de-interlacing the top field are:

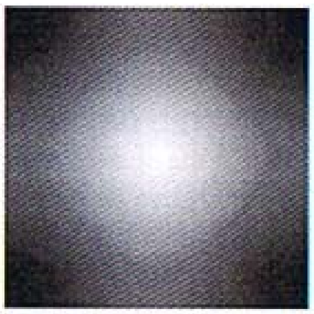
$$h_{fm}(y,t) = \begin{cases} 1 & (y,t) = (0,0) \\ 0 & Otherwise \end{cases} \quad \text{and} \quad H_{fm}(f_y, f_t) = 1 + \frac{1}{2} e^{-j2\pi\Delta_t f_t}$$

But this filter is asymmetric and complex, causing implementation issues. Instead, "**field averaging**" can be used similar to the line averaging case. The filter function and the frequency response will be:

$$h_{fa}(y,t) = \begin{cases} 1 & (y,t) = (0,0) \\ 1/2 & (y,t) = (0, \Delta_t), (0, -\Delta_t) \\ 0 & Otherwise \end{cases} \quad \text{and} \quad H_{fla}(f_y, f_t) = 1 + Cos(2\pi\Delta_t f_t)$$

which is shown below (left).





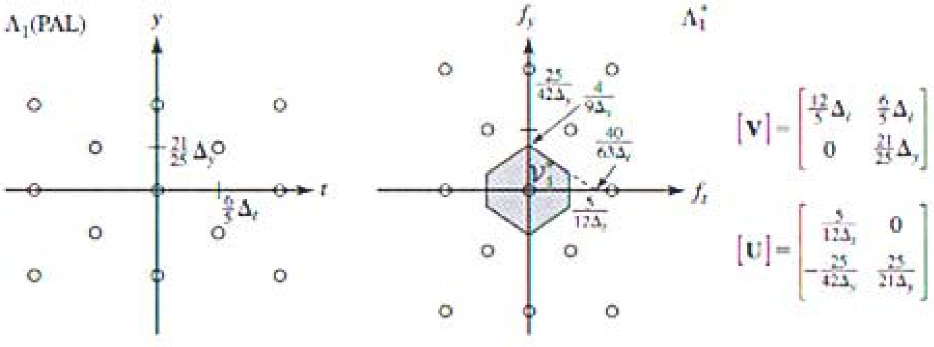
Finally, to compromise between spatial and temporal artifacts from using one-sided averaging, a technique combining the two has been developed and it is called "**line and field averaging**," which uses: $D = (C + E + K + R)/4$. The filter and the frequency response are given by:

$$h_{lfa}(y,t) = \begin{cases} 1 & (y,t) = (0,0) \\ 1/4 & (y,t) = (\Delta_y, 0), (-\Delta_y, 0), (0, \Delta_t), (0, -\Delta_t) \\ 0 & Otherwise \end{cases}$$

$$H_{lfa}(f_y, f_t) = 1 + Cos(2\pi\Delta_y f_y) + Cos(2\pi\Delta_t f_t)$$

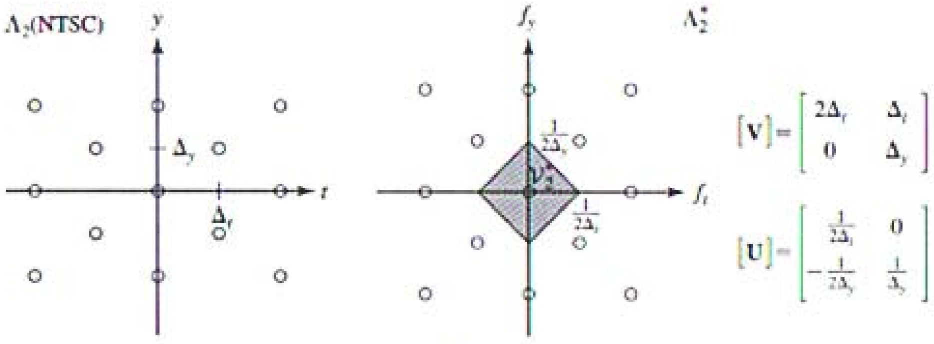
which is shown also above (right).

PAL-to-NTSC Conversion example: This process is significantly more difficult than the de-interlacing problem since the temporal and vertical sampling rates are not integer multiples of each other. We need to come up with a Λ_3 that contains both Λ_1 and Λ_2. By observing $F_{y,1} = 625$ and $F_{y,2} = 525$ yields $F_{y,3} = 13{,}125$. Also, $F_{t,1} = 50$ and $F_{t,2} = 60$ results in $F_{t,3} = 300$. These require a corresponding rectangular lattice such rates as shown in part c below.



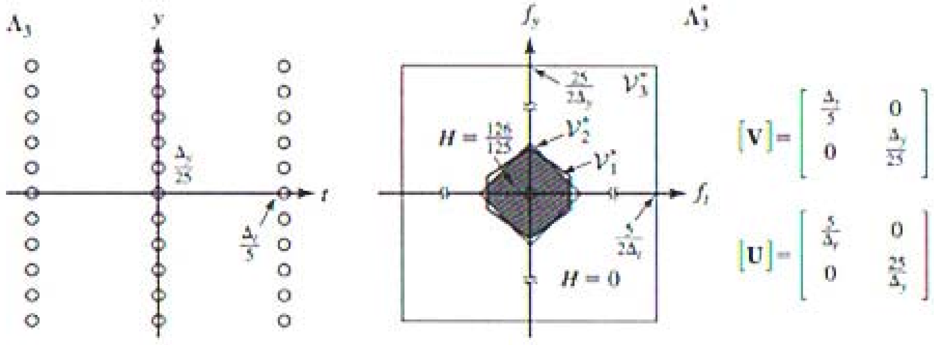
Λ_1(PAL)
y
t
$\frac{21}{25}\Delta_y$
$\frac{6}{5}\Delta_t$
f_y
Λ_1^*
f_t
$\frac{25}{42\Delta_y}$
$\frac{4}{9\Delta_t}$
$\frac{40}{63\Delta_t}$
$\frac{5}{12\Delta_t}$
V_1^*
$[\mathbf{V}] = \begin{bmatrix} \frac{12}{5}\Delta_t & \frac{6}{5}\Delta_t \\ 0 & \frac{21}{25}\Delta_y \end{bmatrix}$
$[\mathbf{U}] = \begin{bmatrix} \frac{5}{12\Delta_t} & 0 \\ -\frac{25}{42\Delta_y} & \frac{25}{21\Delta_y} \end{bmatrix}$

(a)



Λ_2(NTSC)
y
t
Δ_y
Δ_t
f_y
Λ_2^*
f_t
$\frac{1}{2\Delta_y}$
$\frac{1}{2\Delta_t}$
V_2^*
$[\mathbf{V}] = \begin{bmatrix} 2\Delta_t & \Delta_t \\ 0 & \Delta_y \end{bmatrix}$
$[\mathbf{U}] = \begin{bmatrix} \frac{1}{2\Delta_t} & 0 \\ -\frac{1}{2\Delta_y} & \frac{1}{\Delta_y} \end{bmatrix}$

(b)



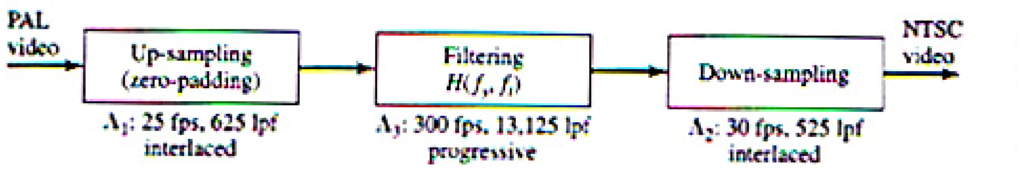
Λ_3
y
t
$\frac{\Delta_y}{25}$
$\frac{\Delta_t}{5}$
f_y
Λ_3^*
f_t
$\frac{25}{2\Delta_y}$
V_3^*
$H = \frac{126}{125}$
V_2^*
V_1^*
$\frac{5}{2\Delta_t}$
$H = 0$
$[\mathbf{V}] = \begin{bmatrix} \frac{\Delta_t}{5} & 0 \\ 0 & \frac{\Delta_y}{25} \end{bmatrix}$
$[\mathbf{U}] = \begin{bmatrix} \frac{5}{\Delta_t} & 0 \\ 0 & \frac{25}{\Delta_y} \end{bmatrix}$

(c)

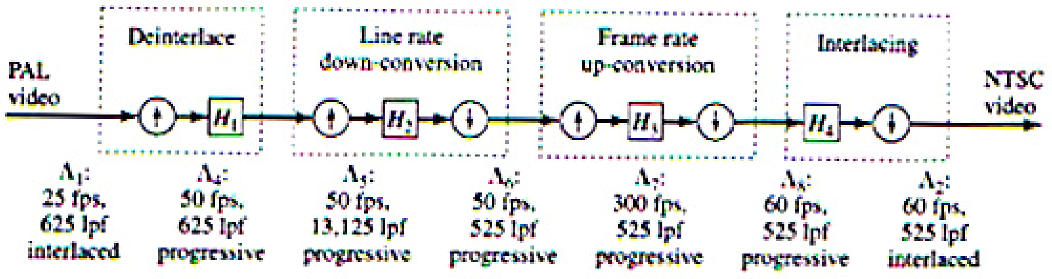
Desired frequency response of the filter over Λ_3 is given by:

$$H_{p\text{-}to\text{-}n}(f_y, f_t) = \begin{cases} d(\Lambda_2)/d(\Lambda_1) = 126/125 & (f_y, f_t) \in V_1^* \cap V_2^* \\ 0 & (f_y, f_t) \in V_3^* \setminus V_1^* \cap V_2^* \end{cases}$$

The ideal conversion filter has the shape shown above part c. This procedure is known as "**direct conversion,**" but more frequently a sequential approach is used in practice as shown below.


PAL video
Up-sampling (zero-padding)
Filtering $H(f_y, f_t)$
Down-sampling
NTSC video
Λ_1: 25 fps, 625 lpf interlaced
Λ_3: 300 fps, 13,125 lpf progressive
Λ_2: 30 fps, 525 lpf interlaced

(a) Direct conversion

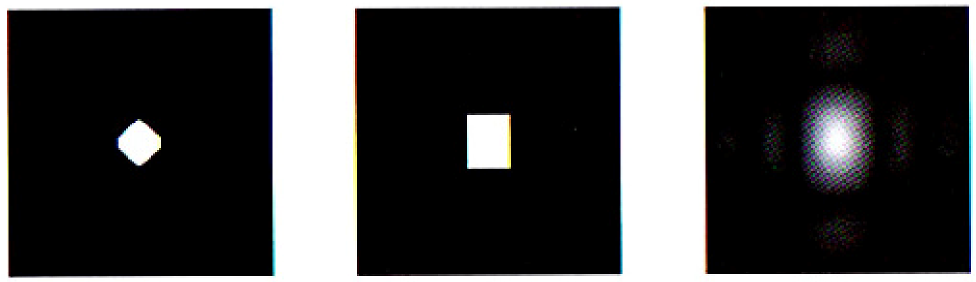

Deinterlace
Line rate down-conversion
Frame rate up-conversion
Interlacing
PAL video
NTSC video
H_1
H_2
H_3
H_4
Λ_1: 25 fps, 625 lpf interlaced
Λ_4: 50 fps, 625 lpf progressive
Λ_5: 50 fps, 13,125 lpf progressive
Λ_6: 50 fps, 525 lpf progressive
Λ_7: 300 fps, 525 lpf progressive
Λ_8: 60 fps, 525 lpf progressive
Λ_2: 60 fps, 525 lpf interlaced

(b) A sequential implementation

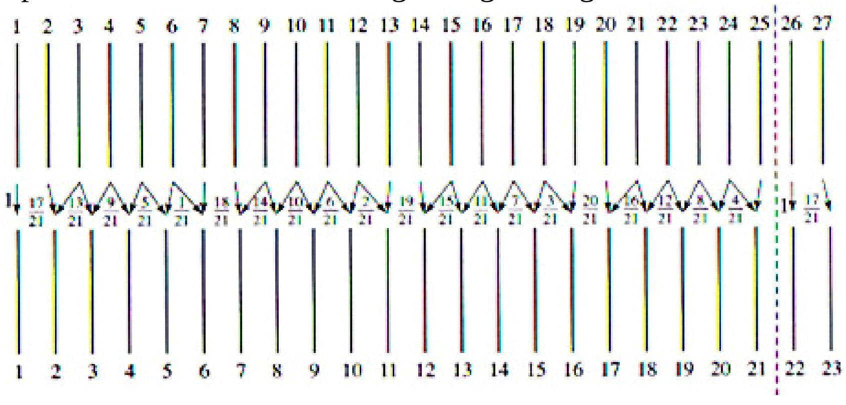
The modified approach is composed of:

(1) De-interlacing each field in the PAL signal into a frame format with 625 lines.
(2) Line rate conversion from 625 to 525.
(3) Frame rate conversion from 50 to 60 Hz.
(4) Splitting each frame into two interlacing fields.

The frequency response of PAL-to-NTSC conversion filters: Ideal filter (left), composite filter corresponding to the concatenation of H_2 *and* H_3 above (middle) and frame-rate up-conversion stage)right.)



Finally, the task of converting 625 lines into 525 (down-conversion) is done by converting every 25 lines into 21 lines and using the two nearest known lines (upper row). For each line in the bottom row, the interpolation coefficient associated with the left neighboring line in the upper row is given in the figure. The interpolation coefficient for the right neighboring line is one minus the left coefficient.


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
1 $\frac{17}{21}$ $\frac{13}{21}$ $\frac{9}{21}$ $\frac{5}{21}$ $\frac{1}{21}$ $\frac{18}{21}$ $\frac{14}{21}$ $\frac{10}{21}$ $\frac{6}{21}$ $\frac{2}{21}$ $\frac{19}{21}$ $\frac{15}{21}$ $\frac{11}{21}$ $\frac{7}{21}$ $\frac{3}{21}$ $\frac{20}{21}$ $\frac{16}{21}$ $\frac{12}{21}$ $\frac{8}{21}$ $\frac{4}{21}$ 1 $\frac{17}{21}$
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

The equivalent vertical filter in Λ_3 is given by:

$$h_v(y) = \begin{cases} 1 - \dfrac{|n|}{2} & \textit{if} \;\; y = n.\Delta_{y,3}; |n| = 0,1,2,\cdots,20 \\ 0 & \textit{Otherwise} \end{cases}$$

The corresponding equivalent temporal filter over the 300-frame grid is given by:

$$h_t(y) = \begin{cases} 1 - \dfrac{|k|}{6} & \textit{if} \;\; t_k = k.\Delta_{t,3}; |k| = 0,1,2,\cdots,5 \\ 0 & \textit{Otherwise} \end{cases}$$

Compared to the desired response (a) the concatenated solution leads to blurred transitions and many ripples in the stop-band. To improve the results “de-blurring” enhancement techniques are normally applied. In addition motion-adaptive interpolation is also used in high-end imaging systems[1].

[1] This segment is summarized from Chapter 4 of Wang, Ostermann and Zhang.