AMERICAN INTERACTIVE MEDIA
A Philips/PolyGram Corporation

AIM Technical Notes

TN #42: Reduction of Flicker in Interlaced Pictures

Written by: Norman Richards, Philips Research Laboratory, Redhill, England

May 4, 1989

Synopsis: Contrary to general belief, the base case CD-I system, the JNMS player, is capable of displaying interlaced pictures in a trivial manner and in a way entirely consistent with the Green Book standard.

The base case CD-I system, the JNMS player, is capable of displaying interlaced pictures; however, I will not elaborate in great detail here. Basically, the method consists of making draw maps in interlaced format that contain the data for odd and even interlaced fields and linking the FCT and LCT with the appropriate interlace parameters set. Then, we use the UCM function that sets the device physically in interlace mode.

Assuming you start with some source material for a fully interlaced picture, once these steps are completed, you are left with a well-defined picture on the screen, with twice the normal vertical resolution. Jaggies on near horizontal line are greatly reduced, and even more important, the big black holes between lines disappear completely. We are back to a real 525 or 625 line system.

That is the good news. The bad news is that in various places in the picture where there is horizontal detail at high contrast, we see pronounced flicker effects, the so-called interline flicker. This is not a CD-I effect. This effect has been with us ever since 1937 when interlaced display was invented as a cunning way of getting twice as many lines for (almost) nothing.

Why are we encumbered with this problem now? There are various reasons. This effect is much more noticeable on still pictures than on moving pictures. Motion appears to have a masking effect on the annoyance level of this phenomenon. It is also dependent on overall brightness level, so the problem has become more acute as the brightness level of displays has increased over the years. It also depends on the vertical resolution of the camera pick-up devices.

Because of this problem, interlace was presented in a low key manner, and, indeed almost escaped the Green Book altogether. Given the advantages of interlace, in particular the removal of the black lines, is it possible to overcome the flicker problem? The short answer is that there is only one complete solution to the problem. Adopt a sufficiently high vertical scan rate, using either interlaced or progressive scanning. The physiological effect of flicker varies inversely at about the fourth power of the scan frequency. Thus, even fairly modest increases in the vertical scan frequency make the problem disappear. Unfortunately, we are stuck with domestic television scanning standards of 50 and 60 Hz for the immediate future.

However, all is not lost. The problem is basically one of small area local flicker. The eye is much less sensitive to small area local flicker than to large area flicker. Thus, if we reduce the contrast between adjacent lines, that is, if we make the information difference between adjacent lines less, the subjective effect of interline flicker becomes much less pronounced. Of course, there are obvious difficulties in achieving this in real pictures. If we have three adjacent interlaced lines, we may make the difference between lines one and two equal to zero,. However, this must result in the maximum difference between lines two and three. Thus, we find that if we use line repeat to fill in alternate lines (which is what we have just described), we are left with objectionable flicker effects. Obviously, symmetry requires some sort of interpolation between lines to smear out the difference between adjacent lines.

In practice, an appropriate filtering operation reduces the observed flicker to a very marked effect. This cannot be the complete solution (consider what must happen if we view a single line with a microscope), but subjectively we can achieve great improvement through these means. An immediate objection is that, although we can remove most of the flicker by vertical filtering of the interlaced image, we have simultaneously halved the vertical resolution. Although filtering deliberately reduces the high frequency information, it does not completely remove it. Even if this were the case, the beneficial effects of removing horizontal jaggies (vertical aliasing) and the elimination of the black holes make the change to interlace worthwhile for many applications. In signal processing terms, we are providing a post sampling anti-alias and interpolation filter by means of oversampling.

In the absence of elaborate subjective experimental data, the reader must accept our word that this vertical filtering appears to give acceptable results. Without arm-twisting, all of the people who have seen the results agree there is substantial subjective improvement.

Vertical filtering algorithm

The object of the vertical filtering algorithm is to reduce the difference between adjacent lines. Filtering is one way to characterize this operation, but smearing the data from one line to another is an equally valid way to describe it. Because the lines are interlaced, nothing is gained by performing the operation on other than adjacent lines. At each horizontal point along the line, we add a fraction **k** of the value from the line immediately above and the same fraction **k** of the value from the line immediately below. However, if we consider the effect of doing this on a uniform picture of identical lines, the amplitude grows by a factor **(1 + 2k)**. To compensate, we subtract the value **2k** from the value of the middle line.

So we have the following:

```
for ( y = 0; y  < MAXY ; y++ )
{
        for ( x = 0 ; x < MAXX ; x++ )
        {
                new_value[y] [x] =      value [y] [x] * (1-2*k)
                         + value [y-1] [x] * k
                         + value [y+1) [x] * k ;
        }
}
```

The more astute reader will notice that we will have difficulty on the first and last line. We are trying to add in elements that do not exist. The possible cures are:

(a) The simplest solution is to alter the limits to 1 and MAXY - 1.
(b) It is better is to have special one sided code loops to deal with the top and bottom lines.
(c) Alternatively, use extra dummy lines top and bottom which have the value zero or repeat the top and bottom lines respectively.

If the picture is whole screen, it will overscan anyway. You may use option (a), but sooner or later you will have to correct this.

If the picture is in a window, we have a separate problem. The horizontal window edge will flicker. Alty Van Luyt has proposed a good solution here. The matte transparency should not change from 0 to 1 abruptly, but should be filtered over (three?) lines. We have not tried this, but it seems to be a good idea. We can write a trivial utility to carry out this operation.

Mixed graphics and natural pictures present a greater problem. Interlace is much more problematical here. In moderation, we might get away with line repeat for the graphics, but only sometimes. We can envision a graphics package that has vertical filtering built in. However, we are not volunteering to write it.

Text also presents a problem. If we have substantial amounts of text to read, we should not run interlaced. We can create an instantaneous invisible switch. However, captions in anti-aliased text may have the filtering built in. (The present anti-aliasing algorithms have to be changed.) If captions are superimposed on pictures, they are essentially part of the picture.

Optimum value for the filtering constant

There is no single correct value for the constant **k**. Ultimately, assignment of a value is subjective. Another complication is that the source material already has different amounts of pre-existing filtering. The spot size of nearly all cameras and flying spot scanners is large enough to overlap adjacent lines. The same effect occurs in the horizontal direction, but the camera manufacturers introduce "aperture filters" to compensate for the effect. They do not compensate for this effect vertically, because (a) a line store is required and (b) enterprising camera manufacturers found that it made the interline flicker worse. The solid stare camera is an exception to this. Provided the optics are good enough, there is no vertical spreading. From our point of view, this includes the flatbed scanner (at least for the lower resolutions).

Practical experiments show that less than **0.1** for the value **k** is a waste of time. Values in the range of **0.2** to **0.33** are probably about right. The last value makes computation simple because the lines each provide a one-third contribution. However, you should write the software for the general case. The other special case is **0.5** which degenerates to interpolation of the two outer lines.