November 2022

TONE-CYCLED HALFTONING FOR Z AXIS PATTERN UNIFORMITY IN 3D MJF

HP INC

Tone-cycled halftoning for Z axis pattern uniformity in 3D MJF

Context of the problem.

The point is that vertical lines appear with the same frequency as the halftoning matrix. What causes the problem is the way to cycle the layers along Z. That is, each of the layers is fine, in the XY planes there are no patterns. But the way the halftoning layers (or ht for short) are painted on top of each other isn't very elaborate. The matrix is translated in the XY plane a fraction of its size. This ensured that each layer was different from the previous and following ones and did not repeat until a certain number of layers. There is a parameter n in the system that refers precisely to that. For a value n, the offset in XY is s/n where s is the size of the halftoning matrix. Usually, the bigger value of n the better. But once n reaches the value of the square root of the size of the matrix, continuing to increase its value becomes counterproductive, since the pattern of the next harmonic begins to be favored.

As the distribution of the pixels is not related to the translation in XY, the vertical pattern produced, how each layer fits with the previous and following layers is merely random. Then, by randomness, different sets of grouped pixels are produced to which a few drops of FA (fusing agent) correspond, creating columns along Z with very little applied contone of FA. The appearance of these columns is determined by the FA contone (the more contone it is, the more difficult it is for columns of pixels to remain unpainted at all). And even if those columns exist, they won't be visible unless they coincide exactly around the surface of the part; Inside the printed piece there are those columns, but they remain hidden. However, this pattern has been appeared consistently during last months. Even if the reasons because it appears look random, the pattern is always there.

The parameter n that we mentioned before is also closely related to the probability with which the error appears; since, for the pattern to be visible, a sufficiently large set of pixels must present a small amount of FA throughout n layers. Thus, the probability of finding in the block of matrices sets of pixels of this type along all layers of size n decreases with order exponentially over n: $o(\frac{1}{p^n})$.

Also, it should be mentioned that these patterns have always existed, since the ht mechanism has remained constant since the first versions of MJF. The reason why it has not been visible until now is because of the existing blooming. The defect we deal with here is caused by a few perfectly Z-aligned pixels without FA. A bloom of a magnitude equivalent to one or two pixels is enough to completely hide the problem. It should be noted that around the year 2021 this problem began to be seen and it was solved by increasing the value of n from 4 to 9, this improvement was sufficient for the following year. However, with the latest improvements in control over the printer, new materials, new physical parameters... in general, a production of parts with less blooming, make these last quantitative improvements insufficient, and a qualitative improvement is necessary. Something better than moving the layers between them and hoping that, randomly, they fit well with each other.

The proposed solution.

The proposed solution consists of, given a ht matrix, we build each new layer by cycling the tones that appear in it. To the entire matrix, we add a fixed value to, from the previous layer, generate a new one, remembering that the values that exceed the maximum tone value t_{max} must start again with 1. In this way, throughout of the layers we will use all the pixels of the matrix along Z before using the initial ones again, creating contone regularity in the Z axis.

Figure 1. Example of using the array with $t_{max} = 255$ and a loop value of 15 in the array values.

Assuming that, in the areas where the pattern becomes visible, it is on the outer surface of the piece, where the contone is (in most cases) constant with a fixed value c, we create a matrix with multiple layers where each is the array with the values incremented by c. That way the pixels used to paint FA are always different on different layers.

$$t_l = ((t_{l-1} + c - 1)\%t_{max}) + 1$$

We must understand that the formula starts simply with $t_l = t_{l-1} + c$, that is, the tone of layer l is the tone of the previous layer with an increment of c. Then, we close the cycle with a module at t_{max} (which are the tones that the matrix has), with a setting of -1 inside the module and $+1$ outside it because the set of representatives is $[1, t_{max}]$ and not the canonical $[0, t_{max} - 1]$.

In our case, we are lucky that our values are $c = 15$ and $t_{max} = 255$. Not only is 15 a divisor of 255, but its quotient is also a prime number. This opens the possibility of using those same 17 layers by making jumps between them of any length, with any number of layers generating the commutative group of order 17. That is, we could use the layers by ordering them one at a time (order: $1, 2, 3, 4, \ldots$) jumping 5 by 5 (order $1, 6, 11, 16, 4, 9 \ldots$), to achieve a contone cycling of 75 tones between two consecutive levels and, after 17 layers, we would have used the 17 generated matrices above before repeating any. The fact that $255/15 = 17$ is prime is what allows us this property. Otherwise, we should be careful, since depending on the cycle generator we choose, it could leave whole fractions of the matrix unused, multiplying the error pattern that we came to eliminate. For instance, in a cycle of 16 elements, if we take the layers jumping 4 by 4, we only are going to use the tones $1, 5, 9, 13$ and 1 again, letting the 75% of the pixel columns without FA. Prime numbers are important.

We have seen that any periodic order of matrices would perfectly cover the entire surface of the piece in a regular way in Z. We must now solve the question of which is the best order. We have at this point the original matrix cycled in 17 steps ($\frac{t_{max}}{c}$ in general). We have 17 matrices, each using as a set of 15 initial tones a different partition of the XY plane. That is, the first 15 tones of all the matrices completely cover the plane, and there is no pixel that belongs to the set of 15 initial tones of two different matrices. It is a partition of the plane. For this reason, the use of these matrices in any order will cause exactly one drop of FA to be painted in each and every one of the columns of the printed piece that have contone 15, out of every 17 layers. We therefore eliminate the possibility that some columns of pixels have more FA than others, we eliminate the possibility

that there are no longer sets or columns of a single pixel that do not have any FA. But it remains for us to answer which is the best order.

Because of how the matrix is built, any given tone works well with the tones around it, particularly the previous tones, since they are usually built-in chromatic order. That is, the 50 tone is built to work well with the previous 49 tones. And that's why we use the matrix tones incrementally, with nested sets of tones. This means that the tone interval $[46, 60]$ will work well with the interval $[31, 45]$ and with the interval $[61, 75]$. We then deduce that, ideally, we should use the arrays in their natural order. That each one has with the previous one exactly a distance of 15 tones. We don't have to worry about the tones that are looped at 255, going around the 0. Because while it's true that the tones $[241, 255]$ aren't meant to work well with the interval $[1, 15]$, they're just the tones that remain in the matrix, the pattern that we are here to correct is not given by the bad transition of one of every 17 layers, it is given by a recursive pattern throughout all the layers, which we have already eliminated.

The importance of the order in which the 17 layers are applied has been proven, since different orders (including random order) were tested. The different orders caused the vertical lines to effectively disappear, but there seemed to be a certain granularity (not roughness) on the vertical surfaces of the pieces. Cycling the tones is enough to eliminate the error that we have been chasing but ordering and aligning the layers that we generate is also essential to avoid other collateral effects, such as noise in the halftoned image.

The matrix with other contones.

We must remember that the array is fully optimized for contone c. We must ask ourselves what the behavior of the FA with different contone values will be.

If the contone used is smaller, even though it is true that the matrix is not optimized and that, with the same idea, we could do something more specific, there will be no problem. We must think that what will happen is simply that there will be tones that will not be used. Since we have generated cycles of $\frac{t_{max}}{c}$ layers, our granularity in Z will be intervals of $\frac{c}{t_{max}}$ (a value of $\frac{1}{17}$ in our particular case), that is, in any column we can make contone jumps $c = 15$ (remember that, with the classic solution, in which the matrix was repeated every n layers, the granularity in Z was contone jumps 64, for example). If we use, for instance, contone $c - 3$, that is, $\frac{c-3}{t_{max}}$ of the total possible contone, the ideal would be to be able to paint one of each $\frac{t_{max}}{c-3}$ layers. What will happen is that we will paint a drop in $c - 3$ of every c columns, and we will not paint anything in the remaining 3 of every c columns; being those 3 columns without using the best possible ones to do it, they are not randomly generated columns without any control, the system is designed so that they are the best possible distributed. That is, the solution converges to the optimum in an order of $o(\frac{1}{d})$ where d is the contone defect with respect to the optimum. There is no relevant problem.

In the case in which we use more contone than expected, an error with respect to the optimum due to excess, in terms of the use of the columns there will be no problem. It can be verified that, for any value of contone used, the difference between any two columns of pixels will always be no more than one drop from each $\frac{t_{max}}{c}$ layers. In our example, with $c = 15$, in the case of using contone levels of, for example, 65, what will happen is that there will be columns with 4 drops

every 17 layers and columns with 5 drops every 17 layers. Being, in addition, the columns of 4 and 5 drops as well mixed as possible.

However, there is a problem, and it is how to choose those 4 or 5 drops. It will happen that, in each column, all the FA drops are painted in consecutive layers, they will not be well distributed in Z. In a column with 5 drops there will be 5 consecutive layers with FA and then there will be 12 consecutive layers without FA. This happens because the difference between each 2 layers is 15 shades of contone. In our example of contone 65, in a pixel that, for instance, in the initial matrix has a value of 8, in the first layers it will take on the values $8, 23, 38, 53, 68, \ldots$ That is, we will paint the first 4 pixels of that column and not It will be until the fifth layer, in which the value of the matrix is 68, that we no longer paint FA. And it will not be until the matrix turns around after 17 layers that it will not return to the value 8, leaving 13 consecutive pixels of that same column without FA.

The way around this is to use the arrays not in their natural order, the difference between one and its next being 15 tones. Instead, use the matrices with jumps of 5 in 5 layers, so that the difference between two consecutive matrices is $15 \cdot 5 = 75$ tones (which is a sufficient value for most of the contons used inside the piece). So that each layer, start painting FA in the closest possible tone to the one the previous layer finished using, and always erring by excess in the jump of tones in consecutive layers, not by default. That is, bring the pitch jump closer to the upper contone usage limit, not the lower limit.

It is, we should use a jumps generator g with the value:

$$g = \left\lceil \frac{c \cdot layers}{t_{max}} \right\rceil.$$

It is, for a contone $c = 80$, with 17 layers, and the max tone $t_{max} = 255$, is recommended jump the layers 6 by 6, because

$$\left\lceil \frac{80 \cdot 17}{255} \right\rceil = \lceil 5'33 \rceil = 6$$

generating a layer cycling in which each pair of tones in the same position in consecutive layers has difference of 90.

Disclosed by Victor Diego and Rita Roca, HP Inc.