Arithmetical approach to refresh rate selection for judder effect minimization

Marc de Courville, Karine Gosse and Marilou de Courville

March 13, 2021

1 Introduction and problem statement

The goals of this contribution is to provide an algorithm based on integer arithmetic to select the best screen refresh rate belonging to a subset of values to minimize the judder effect for a given frame rate of the video displayed. The optimization metric used is to avoid uneven image duration displayed on the screen or maximize the number of uneven image durations so that it is not noticeable for the video watcher.

The objective is to provide means to perform adaptive refresh rate based on video frame rate in video players.

Note that, in this study, any specific TV motion compensation algorithm is supposed to be disabled.

A mis-alignement of the frame rate with respect to the refresh rate of the screen displaying the video results in a non smooth video playback (with jerky movement) caused either by frame drops or un-even picture duration. This is referred as "judder effect".

Judder effect results from the historical fragmentation in terms of analog TV broadcasting standards (cf. https://en.wikipedia.org/wiki/List_of_broadcast_video_formats).

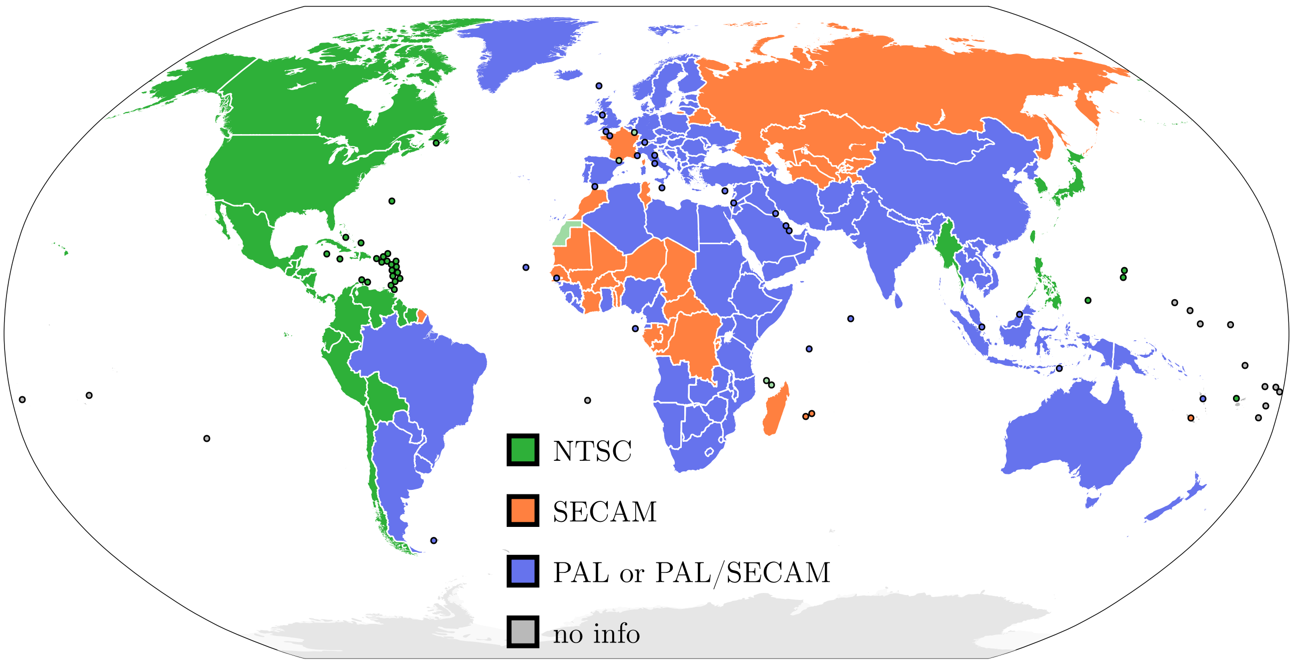

Figure 1: TV broadcasting standards used over the world

Most of the movie industry produces content in 24p (24 frames per second or fps). However in order to be viewed on the TV, content is edited and transformed to compatible fps to match old analog TV sets refresh rate. For instance, in France 50i is used with 50Hz refresh rate and in the US 24p content is transformed to match NTSC standard by slowing down the fps to 24000/1001. Possible mismatch between native refresh rate of the screen and frame rate of the video to be played results in judder. A way to mitigate this effect is to pick the best screen refresh rate available to minimize perceived judder effect. When it comes to perception everything is relative but in this contribution, an arithmetical approach to uneven frame maximization is presented when no multiple of the frame rate is available as refresh rate.

2 Notations

Let f be the refresh rate (in Hertz) of the screen on which the video needs to be displayed.

Let r be the frame rate (in fps: frame per second) of the video to be played.

Common video frame rates are: $f \in \{23.976, 24, 25, 29.97, 30, 48, 50, 59.94, 60\}$ in fps.

Typical screen refresh rates are $r \in \{24, 25, 30, 48, 50, 60, 90, 100, 120, 240\}$ in Hz (s^{-1}).

Note some of the frame rates or refresh rates are not integer due to legacy broadcast constraints where the following frame rates are in reality fractional: $\frac{24000}{1001} \approx 23.976$, $\frac{30000}{1001} \approx 29.970$, and $\frac{60000}{1001} \approx 59.940$.

Thus in what follows, frame and refresh rates are considered per 1001 seconds (by simple multiplication) in order to reduce the problem to be tackled from an integer arithmetic standpoint. Using this trick: $f' = 1001 \times f$ and $r' = 1001 \times r$:

- $f' \in \{24000, 24024, 25025, 30000, 30030, 48048, 50050, 60000, 60060\}$ in frame per 1001 seconds;
- $r' \in \{24024, 25025, 30030, 48048, 50050, 60060, 90090, 100100, 120120, 144144, 240240\}$.

In the sequel, modulo operation is denoted by $a \equiv b[c] \iff \exists k \in \mathbb{Z} \,/ a - b = k \times c$.

3 Proposed algorithm

For a given screen resolution and video frame rate f' to display, the algorithm proposed to minimize the judder effect is to select among all the available refresh rates r' allowed by the display screen the one optimizing the following metrics through two steps:

- select highest r' multiple of f': r' matching $r' \equiv 0[f']$;
- otherwise select highest $r' > f'$ maximizing the number of uneven image durations (referred in the sequel as glitches).

This strategy is assumed to be the less disturbing strategy from a user perception perspective.

Note that perfect decimation is not part of this algorithm i.e. selecting highest r' for which r' is a multiple of f' i.e. matching $f' \equiv 0[r']$. To illustrate this choice, let's take a video at 48 fps and a screen capable of $\{24, 30, 50, 60\}$ Hz refresh rates. There is no refresh rate matching a multiple of 48. Selecting 24Hz, i.e. operating a perfect factor two decimation would result in even image duration but would yield frame drops. This strategy has been tested and results in very bad viewer perception with a non smooth video viewing experience and jerky movement. For 48fps, 60Hz is better suited for a good viewing experience. This is the reason why such strategy is discarded.

Thus judder effect can be generated either by frame drop or by uneven frame display duration. In the algorithm proposed, it is assumed that refresh rates higher than video frame rate exist and thus only uneven frame display duration need to be addressed. Uneven displayed frame duration is caused by a misalignment between the duration between two screen refresh and the video frame duration. The image displayed on the screen is the one sampled at the start of the refresh rate time period as illustrated in figure (2) on which we see that the duration ratio between even and odd frames is $3:2$ (a.k.a. $3:2$ pulldown).

3.1 Number of glitches quantification

In order to tackle the problem from an integer arithmetic, let reduce the problem to one with a refresh rate and frame rate that are coprime (mutually prime). In order to achieve this let's compute $\gcd(r', f')$ and estimate the number of glitches with frame rate $a = f' / \gcd(r', f')$ and refresh rate $b = r' / \gcd(r', f')$. Based on the above definitions, $\gcd(a, b) = 1$.

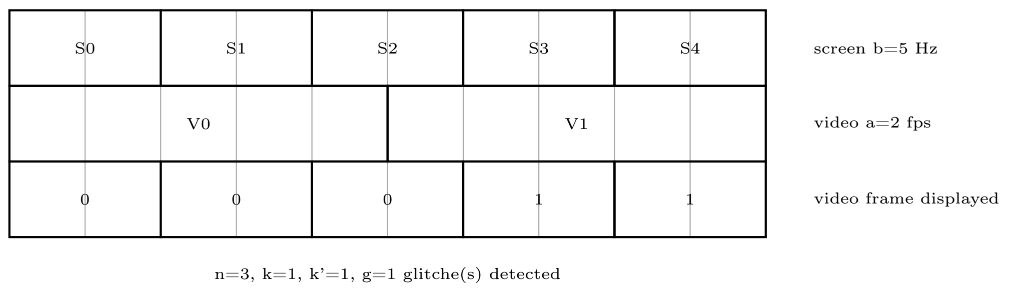

Figure 2: judder glitches illustration, 3:2 pulldown for $r = 60 = 12 \times 5$Hz and $f = 24 = 12 \times 2$fps

From now on, the following assumptions are made: $a < b$, $\gcd(a, b) = 1$.
In order to carry out all the analysis, it is convenient to adopt a time resolution of $\frac{1}{a \times b}$.
With this convention, the following observations can be made:

- during $a \times b$ time slots, the screen is refreshed b times and a video images are displayed
- one video frame lasts b time slots
- the duration between two refreshes is of a time slots
- there are a video image transitions, in $a \times b$ time slots of duration $\frac{1}{a \times b}$;
- there are a video image transitions in b blocks of a time slots (monitor refreshes);
- the frame displayed during kth refresh is the $\lfloor \frac{k \times a}{b} \rfloor$th one

Since $\gcd(a, b) = 1$, the set of remainders in the euclidian division of multiples of a by b spans over all integers from 0 to $b-1$, i.e. $\{k \times a[b],\ k \in \mathbb{Z}\} = [\![0; b-1]\!] = [0; b-1] \cap \mathbb{R}$ (cf. proposition 3.2).

One can notice that the longest repeat sequence is the first one and lasts $n = \lfloor \frac{b}{a} \rfloor + 1$ time slots. The other repeat sequence lasts $n - 1 = \lfloor \frac{b}{a} \rfloor$.

This uneven time block sizes creates the judder effect each of them creating a visual glitch. Let determine the number of perceived glitches.

Only two repeat sequence duration exist: n and $n-1$ covering the entire refresh time and all the video frames are displayed, that is to say:

$$\begin{cases} k \times n + k' \times (n-1) = b \\ k + k' = a \end{cases} \iff \begin{cases} k = b - (n-1) \times a \\ k' = a \times n - b \end{cases}$$

The number of glitches g perceived in $a \times b$ time slots are thus:

$$g' = \min(k, k') = \min\left(b - (n-1) \times a, a \times n - b\right)$$

Denoting by $d = \gcd(1001 \times r, 1001 \times f)$, the total number g of glitches per 1001s is thus given by:

$$g = d \times \min\left(\frac{1001 \times r}{d} - \lfloor \frac{1001 \times r}{1001 \times f} \rfloor \times \frac{1001 \times f}{d}, \frac{1001 \times f}{d} \times \left(\lfloor \frac{1001 \times r}{1001 \times f} \rfloor + 1\right) - \frac{1001 \times r}{d}\right)$$

$$g = 1001 \times \min\left(r - \lfloor \frac{1001 \times r}{1001 \times f} \rfloor \times f, f \times \left(\lfloor \frac{1001 \times r}{1001 \times f} \rfloor + 1\right) - r\right)$$

$$g = 1001 \times \min\left(r - \lfloor \frac{r}{f} \rfloor \times f, f \times \left(\lfloor \frac{r}{f} \rfloor + 1\right) - r\right)$$

One can notice that $r - \lfloor \frac{r}{f} \rfloor f = \text{mod}(r, f) = r\%f$ i.e. the remainder in the division of r by f which yields that the number of glitches per 1001 seconds finally is given by:

$$g = 1001 \times \min\left(\text{mod}(r, f), f - \text{mod}(r, f)\right)$$
$$g = \min\left(\text{mod}(r', f'), f' - \text{mod}(r', f')\right)$$

3.1.1 Resulting proposed algorithm

Let assume that when there is no refresh rate multiple of the frame rate that the best strategy from a viewer perception standpoint is to maximize the number of glitches per second so that they are evenly distributed. This correspond in trying to avoid isolated glitches that would be more noticeable by the viewer.

The above computations can be summarized in the following proposed algorithm to minimize the judder perceived effects:

Algorithm 1: Determine best refresh rate to select to minimize judder effect

```
Input  : $f$ frame rate, $r[]$ array of possible refresh rates
Output: refresh rate
1  $f' \leftarrow 1001 \times f$;
2  $r'[] \leftarrow 1001 \times r[]$;
3  $r'_o \leftarrow 0$;
4  $g_o \leftarrow 0$;
5  for $r'$ in $r'[]$ do
6      if $\text{mod}(r', f') = 0$ then
7          if $r' > r'_o$ then
8              $r'_o \leftarrow r'$;
9          end
10     end
11 end
12 if $r'_o \neq 0$ then
13     return $r'_o/1001$
14 end
15 for $r'$ in $r[]$ do
16     if $r' > f'$ then
17         $k \leftarrow \text{mod}(r', f')$;
18         $k' \leftarrow f' - k$;
19         $g \leftarrow \min(k, k')$;
20         if $g > g_o$ then
21             $g_o \leftarrow g$;
22             $r'_o \leftarrow r'$;
23         end
24     end
25 end
26 return $r'_o/1001$
```

3.2 Annexe: proofs

This subsection provides some proofs of the statements used in the document.

In the series of euclidian division of $k \times a$ by b, $k \in \mathbb{N}$, let's denote by $(q_k)_{k\in\mathbb{N}}$ and $(r_k)_{k\in\mathbb{N}}$ the corresponding series of quotient and remainders:

$$\begin{aligned} k \times a &= q_k \times b + r_k,\ q_k \in \mathbb{N} \text{ and } 0 \le r_k < b \\ q_{k+1} &= q_k \text{ if } r_k < b - a \end{aligned}$$

Proposition 3.1. *In the series of remainders of the euclidian divisions of $k \times a$ by b, $k \in \mathbb{N}$: all the remainders are reached.*

Proof. $(r_k)_{k\in\mathbb{N}}$ is periodic of period b since $b|(a \times b)$, $b \times a = q_b \times b + 0$. Now let prove by contradiction that for $0 \le k < k' < b$ $r_k \ne r_{k'}$. Assume that (k, k') can be found such as $0 \le k < k' < b$ and $r_k = r_{k'}$. This means that:

$$\begin{aligned} k \times a &= q_k \times b + r_k \\ k' \times a &= q_{k'} \times b + r_k \\ \implies (k' - k) \times a &= (q_{k'} - q_k) \times b \end{aligned} \tag{1}$$

Since $\gcd(a, b) = 1$, applying Gauss theorem to equation (1) shows that $b|(k' - k)$. But $0 \le k < k' < b$ means that $0 < k' - k < b$ and the only possibility is that $k' - k = 0$ which is in contradiction with the hypothesis. Hence, $Card\left(\{r_k,\ k \in [\![0; b-1]\!]\}\right) = b$, which shows that all possible remainders in the division of $k \times a$ by b are reached. □

Proposition 3.2. *The quotient of the euclidian division of $k \times a$ by b is the frame picture displayed during the kth screen refresh.*

Proof. Let assume that at $t = 0$, the screen top of page refresh and the start of the video frame are aligned. Let divide the time line in time slots of duration $\frac{1}{a \times b}$ which consists in the time granularity used to define block sizes. Denote respectively by $(q, r) \in \mathbb{N}^2$ the quotient q and remainder r in the euclidian division of b by a, i.e. $b = q \times a + r$ where, $0 \le r < a$ and $q = \lfloor \frac{b}{a} \rfloor$.

It is thus clear that the smaller number of size a blocks, until a size b block boundary is crossed is $\lfloor \frac{b}{a} \rfloor + 1$. The time overlap on the next size b block is thus $0 < a - r \le a$.

This analysis is valid at each block i.e. that the possible overlap with next block when stacking size a blocks is smaller or equal to a.

Based on this analysis, only two image durations (i.e. longest stacking of a blocks before overlapping the next b size block) are possible: $q = \lfloor \frac{b}{a} \rfloor$ or $q + 1 = \lfloor \frac{b}{a} \rfloor + 1$. □

4 Annexe: some common glitches analysis

The following glitches analysis are representing real cases:

- $f = 48 = 12 \times 4$fps and $r = 60 = 12 \times 5$Hz: 12 glitches every second;
- $f = 24 = 6 \times 4$fps and $r = 30 = 6 \times 5$Hz: 6 glitches every second;
- $f = 24 = 12 \times 2$fps and $r = 30 = 12 \times 5$Hz: 12 glitches every second;
- $f = 25 = 5 \times 5$fps and $r = 60 = 5 \times 12$Hz: 10 glitch every second;

Common video frame rates are: $f \in \{23.976, 24, 25, 29.97, 30, 48, 50, 59.94, 60\}$ in fps.

Typical screen refresh rates are $r \in \{24, 25, 30, 48, 50, 60, 90, 100, 120, 240\}$ in Hz (s^{-1}).

Note some of the frame rates or refresh rates are not integer due to legacy broadcast constraints where the following frame rates are in reality fractional: $\frac{24000}{1001} \approx 23.976$, $\frac{30000}{1001} \approx 29.970$, and $\frac{60000}{1001} \approx 59.940$.

Thus in what follows, frame and refresh rates are considered per 1001 seconds (by simple multiplication) in order to reduce the problem to be tackled from an integer arithmetic standpoint. Using this trick: $f' = 1001 \times f$ and $r' = 1001 \times r$:

- $f' \in \{24000, 24024, 25025, 30000, 30030, 48048, 50050, 60000, 60060\}$ in frame per 1001 seconds;
- $r' \in \{24024, 25025, 30030, 48048, 50050, 60060, 90090, 100100, 120120, 144144, 240240\}$.

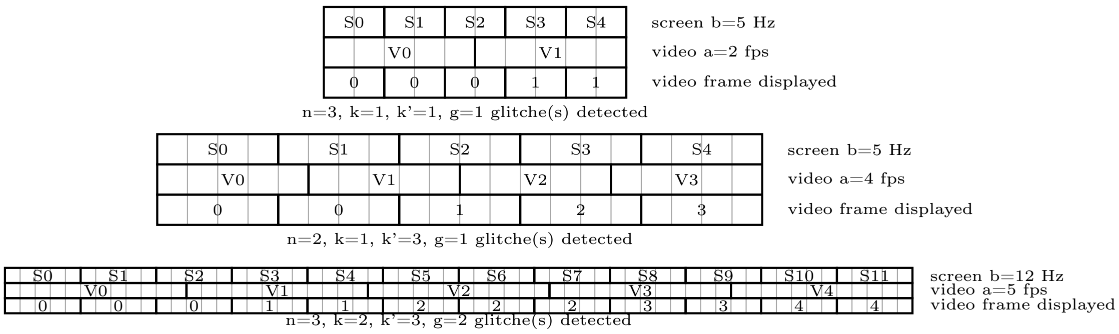

Figure 3: judder glitches quantifications (figures generated automatically)

fps/Hz	1001×fps/1001×Hz	23.976 24000	24 24024	25 25025	29.97 30000	30 30030	48 48048	50 50050	59.94 60000	60 60060
24	24024	24								
25	25025	1,025	1,001							
30	30030	6,030	6,006	5,005	30					
48	48048	48	0	2,002	11,952	12,012				
50	50050	2,050	2,002	0	9,950	10,010	2,002			
60	60060	11,940	12,012	10,010	60	0	12,012	10,010	60	
90	90090	5,910	6,006	10,010	90	0	6,006	10,010	29,910	30,030
100	100100	4,100	4,004	0	10,100	10,010	4,004	0	19,900	20,020
120	120120	120	0	5,005	120	0	24,024	20,020	120	0
144	144144	144	0	6,006	5,856	6,006	0	6,006	24,144	24,024
240	240240	240	0	10,010	240	0	0	10,010	240	0

Table 1: number of glitches per 1001 seconds