Supplemental Material for Variable Frame Timing Affects Perception of Smoothness in First-Person Gaming

DEVI KLEIN, NVIDIA, USA and UC Santa Barbara, USA
JOSEF SPJUT, BEN BOUDAOUD, and JOOHWAN KIM, NVIDIA, USA

1 HELPFUL TERMS

Since we use a large number of abbreviations and some jargon in the main paper, this section contains a list of acronyms that may be helpful when reading the paper.

- **CFT**: Constant Frame Timing - the standard assumed operating model for gaming is frames which complete and are displayed at a constant rate.
- **FPS**: First-Person Shooter - a genre of game where players tend to aim at targets while playing and press a button to fire their weapon. On PCs, players typically control this type of game by aiming with a mouse and using the left click to fire the weapon, making the aiming task similar, but somewhat distinct to 2D cursor-based aiming tasks. In particular, the camera is rotated in the 3D world in an FPS game, causing the cursor to appear to move in the world while remaining fixed on screen. Additionally, the camera projection transformation causes the motion of the world to be somewhat non-linear across the screen, and objects off screen can be revealed as the camera rotates.
- **Frame Rate**: Under the assumption that all frame times are equal, the frames are completed, or presented on screen, with a single frame rate. Common frame rates include 60 Hz (PC and TV), 24 Hz (Movies), 30 Hz (many web videos), 120, 144, 240, 360, 480, and 500 Hz (high end PC and esports games).
- **Frame Time**: The amount of time it takes for a frame to be generated after the previous frame was completed.
- **VFT**: Variable Frame Timing - the phenomenon by which frames are completed after varied durations of time as opposed to a regular frame time.
- **VRR**: Variable Refresh Rate - the ability of some monitors (such as G-SYNC) to display frames as they are completed by the device rendering them (often a GPU).
- **VSync**: When VSync is enabled in an application, newly completed frames are only sent to the display in sync with a CFT dictated by that display. This removes screen tearing artifacts, but causes frames to be repeated if no new frame is completed in time (stutter).

2 MEAN FRAME TIME FOR A SEQUENCE

For a pair of frames, the frame time is the time between the first frame and the second frame finishing. Since VRR displays will draw the second frame on screen when it arrives, the first frame will be shown for that duration. As more an more frames complete, each pair of frames produces another frame time, and in the limit, the number of frames and the number of frame times converge to the same number (alternatively you can assume the first frame just initializes the display, and only count frames beyond the first).

$$\mu_{\text{Frame}} = \frac{\sum(t_i)}{N} \tag{1}$$

Authors' addresses: Devi Klein, devik@nvidia.com, NVIDIA, USA and UC Santa Barbara, USA; Josef Spjut, jspjut@nvidia.com; Ben Boudaoud, bboudaoud@nvidia.com; Joohwan Kim, sckim@nvidia.com, NVIDIA, USA.

One might naturally store each of these frame times in a list and compute statistics over that list, most commonly the mean. This method, which we refer to as the index-based frame time mean (μ_{Frame} – see Equation 1), results in a mean that is noticeably below the center of a sine wave generated to evenly spread out over the runtime of the sequence (Fig. 1). The reason for this difference is because the x axis is the elapsed time, and the y axis of frame time interacts with the x axis, causing the shorter frame times (lower on the y) to be weighted more heavily as that part of the curve is sampled more frequently.

$$\mu_{\text{Time}} = \frac{\sum t_i * t_i}{\sum t_i} \tag{2}$$

The way we choose to compensate for this bias is to compute a time-weighted mean (μ_{Time} – see Equation 2), which multiplies the duration of time that each frame time is displayed. The most notable benefit of this approach for sine waves is that the amplitude of the sine is the same distance in each direction from this mean. While the perceptual results suggest that the longer running frames may have more impact on perception than the shorter running ones, this time-weighted mean is objectively between the min and the max for sinusoids and we recommend using this approach for arbitrary sequences as an objective mean frame time. In the future, a perceptually weighted mean may become more important than the time-weighted mean, but we believe for data and studies are needed to propose such a measure.

$$\mu_{\text{fps}} = \frac{N}{\sum(1/t_i)} \tag{3}$$

The final mean in Figure 1 is the mean frame rate (μ_{fps} – see Equation 3) and we recommend agaist ever using it and only include it here for completeness. This type of measure is likely a mistake if it ever gets performed because it converts each frame time into a frame rate ($\frac{1}{\text{FT}}$) and then computes the mean over the series of frame rates. Then if a frame time is the desired measure, that frame rate gets inverted again ($FT = \frac{1}{\text{fps}}$). This method notably biases even more heavily toward the fast frames.

3 GENERATING A FRAME TIME SEQUENCE OVER TIME

In deciding which type of variable frame timing (VFT) to use we had many different candidates. Initially we considered a "1:N stutter" scheme; in this model a series of N frames occur at constant frame timing, followed by a single (or small set of) frame(s) that is longer/shorter as desired. This creates rapid change in frame timing that emulates a single performance hitch or unpredictable rendering event which represents situations often encountered in real games. However, this 1:N model has several properties which make it challenging to study. First, the change in frame time is abrupt, often more noticeable than a smoothly varying frame time. While representative of some real-world frame time changes, this approach may exaggerate results for our ability to measure more subtle or smooth changes in frame time. Second, since a single (or small set of) frame times are all that represent the dynamics of the frame time, it creates a time-selection process in both perception and performance experiments. If the user is most focused on perception, or completes critical aiming tasks, in the time window away from a 1:N stutter, the impacts of variable frame timing may alternate between almost unnoticeable to completely dominant. For similar reasons we avoided using pseudo-random frame time stimuli; it would be difficult to tell if perceptual and performance outcomes resulted from condition parameters or one-off randomized ordering effects.

Instead we considered other, more smoothly changing, periodic frame time series that incurred more regular change. Within this smoothly changing stimulus regime a variety of periodic signals make sense: sine, triangle, and sawtooth waveforms were all potential candidates. We chose to study a sine-based frame time series as it represents a good

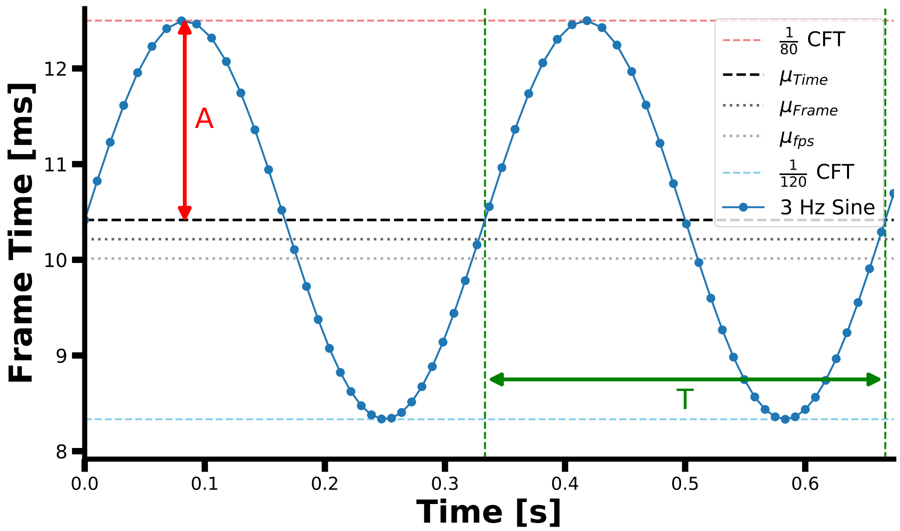

Fig. 1. An example 3 Hz sine wave with amplitude $A = 2ms$, period $T = 0.33s$ and various means plotted. These means were generated with different approaches from the same source data.

trade-off of smoothness to rapid linear change. A sine wave's derivative is continuous everywhere (unlike triangle, sawtooth, or shark fin waves) and includes period of rapid change around its average crossing. We do not expect the results for this sine wave to directly apply to all other types of frame time variability, but it should generalize to a variety of smoothly varying frame timings.

In parameterizing the sine wave stimulus for our conditions we chose to primarily focus on the average frame time (μ_{Time}, weighted by time) and the amplitude (A) or range of variation from this average time. For the purpose of analysis, we compute the amplitude of variation as the absolute value of the difference between the average frame time (μ_{Time}) and the minimum or maximum for a given frame time sinusoid. This experiment does not consider effects of the frequency of this variation, instead we chose to fix the period (T) of frame time variation to 3 Hz or a 333 ms period as a compromise between low enough frequency variation that it would likely be perceived by the user but high enough frequency that many (>10) periods of the frame time variation waveform would fit into a single four second trial. Our approach to sinusoidal frame time variation is demonstrated visually in Figure 1.

3.1 Sampling Strategy

Correctly sampling a waveform of frame time *over time* can be a challenge. If incorrect assumptions about the time-domain variation of the waveform are made the resulting temporal stimulus presented to the user will not have the desired frequency-domain properties. The simplest such assumption is producing frame times that vary by *index* instead of time. As demonstrated in Figure 2 this results in compression of the periods in which lower frame times are displayed and expansion of those in which longer frame times are presented[1]. We avoid this issue by using a time-based sampling strategy wherein the frame time to use for the each sample is calculated based on the time elapsed from the last sample,

[1]This sampling problem is effectively the same as the problem with computing the mean by frames versus the time-weighted mean.

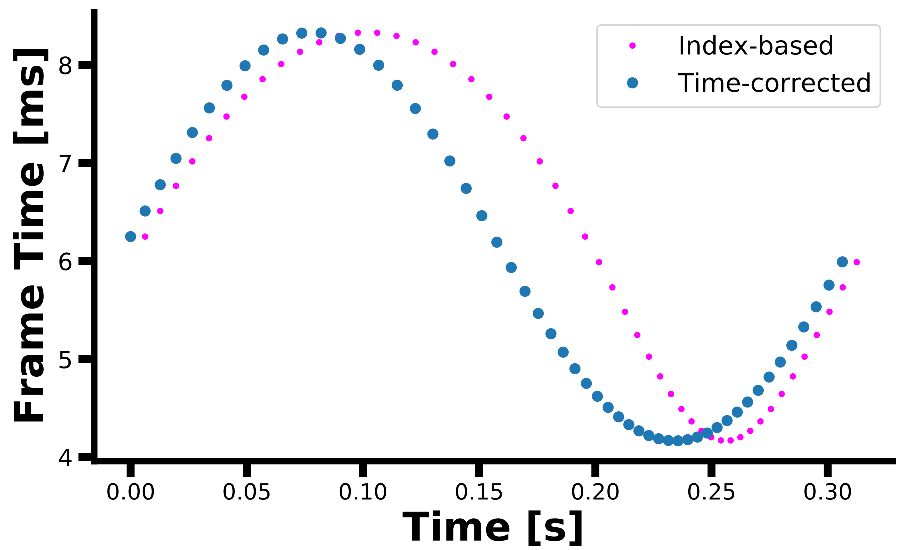

Fig. 2. Comparison of an index-based naïve sampling of a frame time sine wave and a time-correct approach that faithfully reproduces a 3 Hz time domain frame time variation.

resulting in a temporally stable frequency of variation. Listing 1 gives python code to generate a VFT sequence with a sine wave given total duration, min and max frame times and sine wave frequency.

Listing 1. Python to generate a Sine wave sequence for 5 seconds of VFT from $\frac{1}{40}$ to $\frac{1}{30}$ s

```
# Define VFT parameters
DURATION_S = 5          # Total duration of signal to generate in seconds
VFT_FREQ = 3.0          # VFT sine wave at 3 Hz
VFT_MIN_T = 1.0/40.0    # Minimum frame time
VFT_MAX_T = 1.0/30.0    # Maximum frame time

import numpy as np
def getFrameTimes(dur, f, minT, maxT):
    ts = []; fts = []; t = 0
    while t < dur:
      ts.append(t)
      fts.append((np.sin(2*np.pi*f*t)+1.0)/2 * (maxT-minT)+minT)
      t += fts[-1]
    return ts, fts

# t and s now contain a sequence of times (t) and frame times (s)
t, s = getFrameTimes(DURATION_S, VFT_FREQ, VFT_MIN_T, VFT_MAX_T)
```

```
# Optional plotting
import matplotlib.pyplot as plt
plt.scatter(t, s, marker='o')
```

3.2 Sine Waves Used

We used the sampling method described in Section 3.1 to generate the sine waves we used. Listing 2 gives an example of one of these sequences. Figure 3 plots the wave forms over time.

Listing 2. Sine wave sequence for two periods of VFT from $\frac{1}{40}$ to $\frac{1}{30}$ s

```
0.02916667, 0.03134374, 0.03295367, 0.03325759, 0.0320158, 0.02978711,
0.02749811, 0.02582574, 0.02504909, 0.02520957, 0.02624495, 0.02800561,
0.03017935, 0.03219706, 0.03328684, 0.03286648, 0.03106617, 0.02870006,
0.02663444, 0.0253546, 0.0250081, 0.02557961, 0.02697156, 0.02897315
```

4 PILOT STUDIES

In preparing the main studies, we conducted a few pilot studies to help narrow down the conditions of interest. In this section we briefly discuss the most relevant of those pilot studies.

4.1 Sine Wave Frequency

We conducted a brief pilot study considering sine wave frequencies that varied from 2-10 Hz with a handful of conditions in that range. We held the mean and amplitude of frame time variation constant for this pilot study, and only varied the frequency of the sine wave itself. This range of frequencies was selected to ensure enough samples of the sine wave were present, and that it was fast enough that we could expect a user to experience at least a full sine period within the expected task completion time of 500-700 ms. In this study, 4 people participated, including authors and we found only minor variations in the smoothness rating. The sine wave at 3 Hz had a minor advantage in smoothness rating over the others, but we did not run statistical tests for significance. In the end, 3 Hz seemed like a reasonable selection for the main study, and at least not significantly worse than any of the other reasonable choices we considered.

4.2 Frame Time Range

We designed an ran a series of pilot studies for measuring smoothness through a rating system. These included force comparison between paired conditions as well as Likert scales similar to the scale we ended up using in the main study. In these pilots, in which a handful of people who were mostly authors participated, we saw the most significant differences in smoothness ratings from 1/30 to 1/120 constant frame times, and observed the VFT conditions falling in the same range of smoothness ratings while varying somewhat from matched mean constant frame times. Additionally, we saw little difference between forced choice and the Likert scale, except for the saturation of the Likert scale near the lowest frame times (1/240). It may be necessary to use two interval forced choice to get a clearer picture of how things vary when the frame time is below about 5 ms.

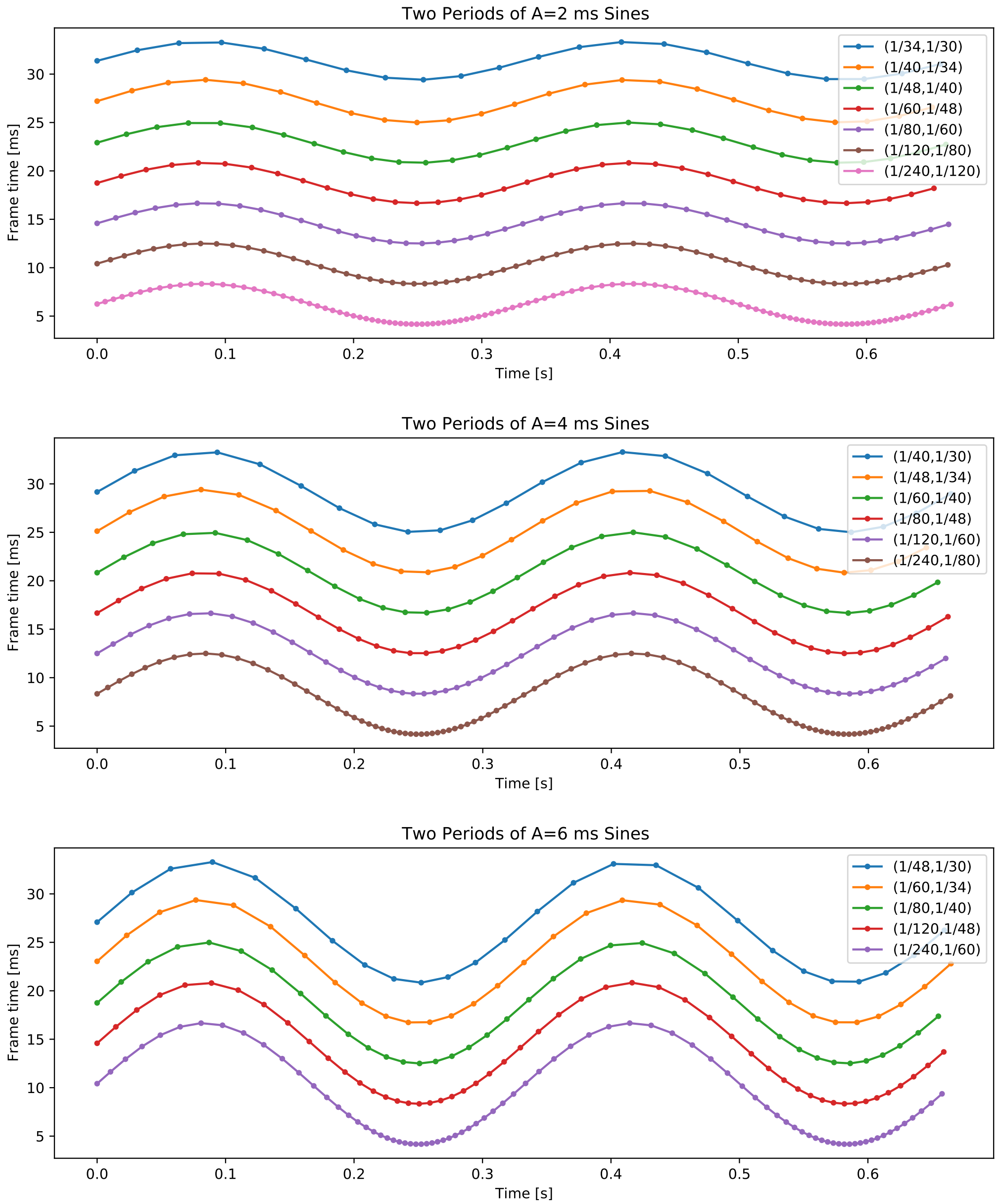

Fig. 3. Two periods of each of the VFT sine waves used in experiment 1. Experiment 2 used a subset of these same sines. The actual waveforms used were long enough to cover the entire runtime of each trial (5 s).

5 EXPERIMENT SUBJECT INFORMATION

We provide detailed subject information for experiment 1 and 2 below in the form of a table. Subjects were recruited based on familiarity with first-person gaming and a (corrected) Snellen acuity of 20/20, these were the only criteria for inclusion in the study.

	Gender		Age			Gaming (hrs/week)				FPS Gaming (hrs/week)				Vision	
Exp #	**M**	**F**	**18-22**	**23-27**	**>28**	**<5**	**5-10**	**10-30**	**>30**	**<5**	**5-10**	**10-30**	**>30**	**20/20**	**Corr.**
1	8	1	3	4	2	1	3	3	2	2	5	1	1	3	6
2	13	2	6	6	3	3	6	5	1	5	8	2	0	8	7

10 total subjects participated in Experiment 1 (1 outlier was excluded from analysis) and 17 total subjects participated in Experiment 2 (2 outliers were excluded from analysis). All subjects either had 20/20 vision or had their vision corrected to 20/20.