Computational Time-Lapse Video

Eric P. Bennett* Leonard McMillan†
The University of North Carolina at Chapel Hill

Abstract

We present methods for generating novel time-lapse videos that address the inherent sampling issues that arise with traditional photographic techniques. Starting with video-rate footage as input, our post-process downsamples the source material into a time-lapse video and provides user controls for retaining, removing, and resampling events. We employ two techniques for selecting and combining source frames to form the output. First, we present a non-uniform sampling method, based on dynamic programming, which optimizes the sampling of the input video to match the user's desired duration and visual objectives. We present multiple error metrics for this optimization, each resulting in different sampling characteristics. To complement the non-uniform sampling, we present the *virtual shutter*, a non-linear filtering technique that synthetically extends the exposure time of time-lapse frames.

CR Categories: I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—Sampling; I.4.3 [Image Processing and Computer Vision]: Enhancement—Filtering

Keywords: Time-lapse, computational photography, video, non-uniform sampling, aliasing, summarization, camera simulation

1 Introduction

Time-lapse is an effective tool for visualizing motions and processes that evolve too slowly to be perceived in real-time. Time-lapse videos are regularly used to capture natural phenomena and to provide artistic cinematic effects. Time-lapse related techniques are also frequently applied to other applications including summarization of films and time-compression of surveillance videos. In this paper, we present a system for generating time-lapse videos with improved sampling, reconstruction, and enhanced artistic control.

Traditional time-lapse methods use uniform temporal sampling rates and short, fixed-length exposures to capture each frame. As is the case with any periodic sampling method, the sampling rate must be sufficiently high to capture the highest-frequency changes, otherwise aliasing will occur. This places an upper bound on the capture interval of the time-lapse output if it is to be free of aliasing artifacts. When shorter versions of the output are desired, the filmmaker is forced to make a tradeoff between aliasing, which exhibits itself as *popping* artifacts in time-lapse videos, missing motions, or pre-filtering, which introduces blurring.

Our approach simplifies time-lapse capture by removing the need to specify a sampling rate and exposure time a priori. This is accomplished through non-uniform sampling of a video-rate input to select salient output frames and non-linear integration of multiple frames to simulate normalized long exposures.

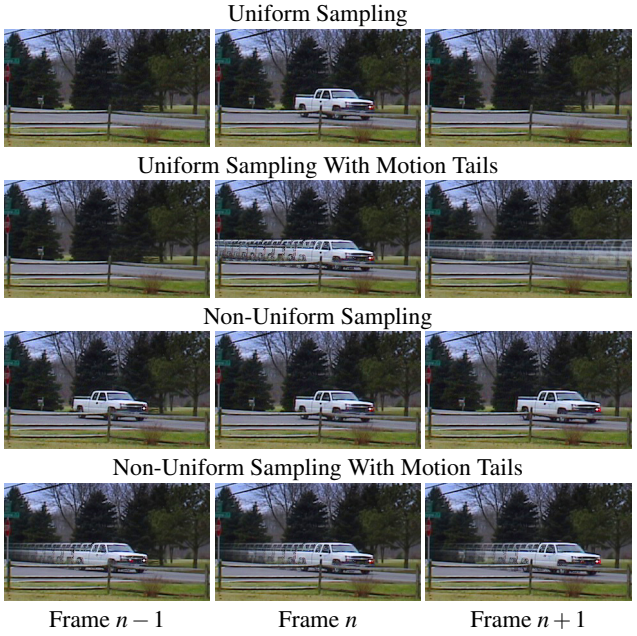

Figure 1: Sequences of three consecutive time-lapse frames that illustrate the sampling issues of traditional time-lapse methods along with our techniques to reduce aliasing artifacts. In a standard uniform sampling, the truck appears in only a single frame, resulting in *popping*. Adding motion tails makes the truck more noticeable across multiple frames. Our non-uniform sampler chooses additional frames for its output containing the truck. Finally, we show a result combining both techniques.

Our first underlying technique chooses the constituent frames in the output time-lapse by calculating an optimal non-uniform downsampling of the video-rate footage. By downsampling after capture, the most important frames can be selected with knowledge of all motions and changes that occur during the video. Our sampling is based on dynamic programming and supports the general set of all error metrics defined between pairs of frames. We discuss error metrics to achieve a variety of effects, including maximizing change, minimizing change, and controlling sampling uniformity.

Our second technique simulates a virtual camera shutter by combining sequential frames together to simulate extended exposure times. This *virtual shutter* acts as a non-linear downsampling prefilter that reduces aliasing artifacts. One virtual shutter setting provides *motion tails* that depict dense motion paths over time similar to the multiple-exposure motion studies of Muybridge [1955]. However, our computational framework enables non-linear combinations not achievable via traditional photography, including "over" compositing, maximum, and median operations.

The key contributions of this work are:

- A method for generating improved time-lapse videos from video-rate footage
- A user-controllable, non-uniform temporal sampler
- A virtual shutter for non-linear exposure time extension

*bennett@cs.unc.edu (Eric P. Bennett is now at Microsoft Corporation)
†mcmillan@cs.unc.edu

2 Prior Work

Our computational time-lapse video methods draw from a wide range of techniques developed for computational photography, video summarization, and video processing. Our work also extends traditional film-based time-lapse techniques, such as those developed for biological microscopy [Riddle 1979] and cinematic applications [Kinsman 2006].

The playback of videos at faster than their original speeds has been investigated by the multimedia summarization community. Video Skimming [Smith and Kanade 1997] looks for short, representative video segments that best tell the story of the video. Segments are chosen based on characteristics including scene changes, camera motion, and audio. The documentary films they target have distinct scenes changes, unlike time-lapse sources. Also, Hua et al. [2003] search for video segments that contain motion between shot boundaries and combine them to match an audio source. Wildemuth et al. [2003] explore how fast videos can be fast-forwarded using uniform sampling while remaining coherent (roughly 64x).

Our use of non-uniform sampling is similar to "Video Summarization By Curve Simplification" [DeMenthon et al. 1998]. They choose a non-uniform temporal sampling based upon simplification of tracked motion-path curves. Our sampling work solves a superset of these formulations and presents a general framework that supports any error metric definable between frame pairs, accommodating a wider variety of time-lapse characteristics. Also, we achieve an optimal minimal-error solution through the use of a dynamic programming-based solution [Perez and Vidal 1994] as opposed to the greedy Douglas-Peucker algorithm [Douglas and Peucker 1973]. Others have also considered the use of pairwise error metrics for jumping between video frames. "Video Textures" [Schödl et al. 2000] looks for transitions within a video that are least noticeable to indefinitely extend playing time. As in our work, dynamic programming is used to find potential jumps. Our solution formulation differs because time monotonically increases in time-lapse and all jumps result in a single output frame.

In "Video Summarization Using MPEG-7 Motion Activity and Audio Descriptors" [Divakaran et al. 2003], short video clips are identified as containing significant motion and are played at real-time speeds and assembled into the final video. Doing so sets a lower bound on the video's duration. Our approach places no constraints on the final duration or playback rates, and the sampling we derive is optimal for the user-specified duration and error metric.

Rav-Acha et al. [2006] specifically address time-lapse videos, allowing events to occur simultaneously and/or out of chronological order, resulting in disjoint image regions combined using 3D Markov fields and arranged using simulated annealing. Their results and techniques differ significantly from ours and can be seen as complementary tools.

Our virtual shutter combines multiple frames into synthetic extended exposures. Computational photography deals with the related problem of combining multiple still images or video frames into a single output frame. "Interactive Digital Photomontage" [Agarwala et al. 2004] allows the user to choose individual parts of a few images to find the best composite image. This fusion is accomplished using both graph-cut [Kwatra et al. 2003] and gradient domain [Perez et al. 2003] algorithms. Similar to our virtual shutter, "Image Stacks" [Cohen et al. 2003] defines methods for automatically selecting and combining image regions using compositing and min, max, median, and low-pass filtering. We extend these ideas with alpha-blending "over" compositing [Porter and Duff 1984] to illustrate the passage of time and by generating a video output with an adaptive sliding temporal window. Our non-linear exposure combination also has similarites to [Bennett and McMillan 2005]. However, that work avoids combining frames with motion, while we now combine frames to illustrate motion and reduce aliasing. Finally, our results are similar to the stroboscopic multiple exposure photographs of Harold Edgerton [Edgerton and Killian 1979] and the motion studies of Muybridge [1955] and Marey [Braun 1995].

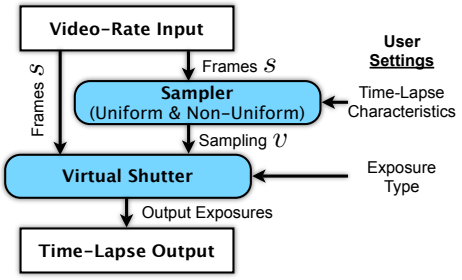

Figure 2: Diagram of the data flow in our computational time-lapse video system. Video-rate footage, s, is input into the sampler, which chooses the sampling, v, that best matches the user's desired duration and characteristics. This sampling determines the times and durations of the virtual shutter's non-linear synthetic exposures.

3 Traditional Time-Lapse Techniques

There are two popular ways to configure a film-based time-lapse setup. The most common employs a device called an *intervalometer* that takes a fixed-length exposure every few seconds, with sampling interval T_s (Eq. 1), also called the time step. Doing so risks missing motions that occur between exposures or undersampling fast motions. A less common approach, for low-light conditions, takes longer exposures throughout the entire time step, increasing motion blur. Motion blur guarantees high-frequency motion will be imaged, but that motion will be blurred and often fades into the background.

$$T_s = \frac{EventDuration}{totalOutputFrames - 1} \tag{1}$$

As digital still photographers have long known, it is preferred to capture at high resolutions then downsample and crop as a post-process, allowing multiple alternatives to be explored non-destructively. We argue that a similar philosophy should be applied to constructing time-lapse videos by capturing at a higher frame-rate than the T_s step dictates, then temporally downsampling as a post-process. Until recently, capturing long videos at high resolutions was impractical due to storage requirements, but it is now possible to store days of video on most commodity PCs.

In computational time-lapse video, time is discrete and is therefore sampled at video-rates into frames. Taking a single short exposure at each time step (*i.e.*, one sample from many samples) is a non-optimal downsampling, potentially generating temporal aliasing. The discrete sampling interval T_d is an re-expression of T_s:

$$T_d = \frac{totalInputFrames}{totalOutputFrames - 1} \tag{2}$$

Considering this uniform downsampling as modulation with an impulse train, the train's phase (the offset of the first sample frame) can dramatically alter the results. Leaving the shutter open for a longer fraction of each time step is analogous to convolving the source signal with a low-pass filter prior to downsampling. While this decreases aliasing, it is not ideal for conveying motion. Our methods provide flexibility both in which frames are chosen and how samples are integrated to simulate an exposure.

Another class of time-lapse systems uses feedback instead of an intervalometer. Using a CCD attached to a computer, the last recorded exposure is compared with the most recent exposure. If they are dissimilar enough, the new exposure is recorded and the process repeated. Similarly, some surveillance cameras use motion detectors to trigger video-rate recording. Our method generates a video of user-specified duration whereas these approaches cannot.

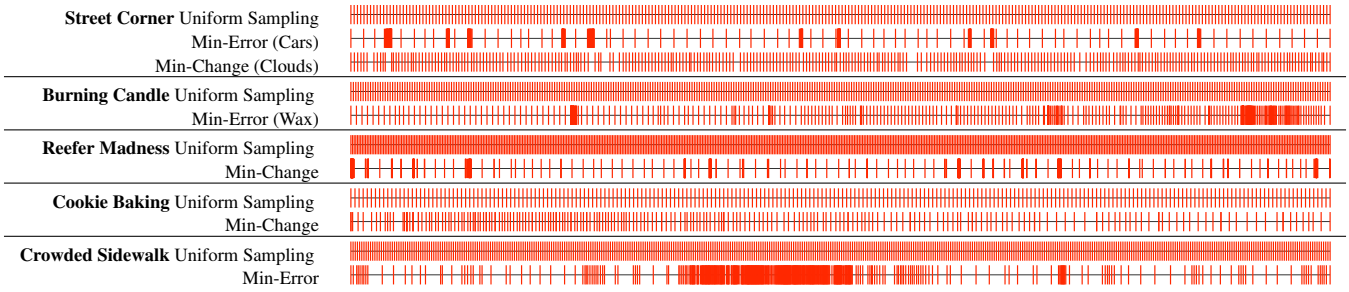

Figure 3: Visualization of the sampling results, where each vertical line represents a sampled frame in v. Samplings using the min-error metric choose the majority of their frames from within periods of change and motion to best approximate the video. Alternatively, the samplings using the min-change metric avoid frames dissimilar to other frames.

4 Non-Uniform Temporal Sampling

In this section, we develop a non-uniform temporal sampler to select the frames of a time-lapse video from a video-rate input to achieve user-specified duration and visual characteristics. This is accomplished using a dynamic programming solution for which a suite of error metrics are presented to generate a variety of outputs.

4.1 Min-Error Time-Lapse Video Sampling

We consider the general problem of finding v, the optimal M frame time-lapse sampling of an N frame video-rate input source, s. Furthermore, we now define the *optimal* v as the time-lapse sampling that retains as much of the original motion and change as possible from s. Note, we use subscripts to index frames and superscripts to index spatial pixels within s, while v is a list of frame indices into s.

To find v, we seek to find the M frames whose piecewise-linearly-interpolated reconstruction best approximates s in the least-squares error sense. This interpolation can be thought of as a *cross-fade movie* where adjacent frames in v are blended together to create the interim frames. The reconstruction error is thus the squared difference between the interpolated video and s. After sampling, the frames in v are played sequentially (without interpolation), resulting in an M frame time-lapse video maximizing motion.

The optimal solution has the lowest total error summed across each of its cross-fade segments. Thus, we define an error metric, $\Delta(i,j)$, that measures the error between the pixels in the the original video-rate input and a cross-fade starting at frame i and ending at frame j (modeled with slope b_{ij}^{xy} and y-intercept a_{ij}^{xy} for each pixel). Because, in the resulting time-lapse output, we omit the interim frames between i and j, $\Delta(i,j)$ can also be thought of as the cost of jumping between frames. We refer to this as our *min-error* metric:

$$\Delta(i,j) = \sum_x \sum_y \left[\sum_{k=i}^{j} \left((a_{ij}^{xy} + b_{ij}^{xy} k) - s_k^{xy} \right)^2 \right] \tag{3}$$

$$a_{ij}^{xy} = s_i^{xy} - b_{ij}^{xy} i \qquad b_{ij}^{xy} = (s_j^{xy} - s_i^{xy})/(j-i) \tag{4}$$

We now seek $\hat{D}(s,M)$, which is the optimal sampling set v, that results in the minimum error reconstruction of s when $M = |v|$. The reconstruction error is the total of the individual $\Delta(i,j)$ errors:

$$\hat{D}(s,M) = min_v \sum_{n=1}^{M-1} \Delta(v_n, v_{n+1}) \tag{5}$$

$\hat{D}(s,M)$ is expressed via the following recursion relationship:

$$D(n,m) = \begin{cases} min_{m-1 \le i \le n-1}(D(i,m-1) + \Delta(i,n)), & n \ge m > 1 \\ 0, & n = m = 1 \\ +\infty, & otherwise \end{cases} \tag{6}$$

$$\hat{D}(s,M) = D(N,M) \tag{7}$$

This recursion is efficiently solved with dynamic programming [Perez and Vidal 1994]. For details of the min-error time complexity and derivation in both 1D and 2D, please refer to [Bennett 2007].

Although this solution produces useful results as-is, we include *shape* controls to modify the $\Delta(i,j)$. This way, the user can affect the impact of particularly low or high errors on the final sampling. Specifically, we define a modified form:

$$\Delta'(i,j) = (\beta \Delta(i,j))^{\alpha} \tag{8}$$

In many cases, α and β are near 1, and adjusting them affects sample clustering. For videos with significant scene or sensor noise an advanced form, with a per-frame threshold term γ, can be used:

$$\Delta'(i,j) = (\beta \times MAX(0, \Delta(i,j) - MAX(0,(j-i-1)\gamma))^{\alpha} \tag{9}$$

4.2 Min-Change Error Metric

The algorithm in the previous section generates optimal time-lapse videos that preserve as much motion and change as possible. Now, we consider *optimal* time-lapse videos as those that avoid including frames that are significantly different from the other sampled frames. This implies choosing similar frames that minimize temporal aliasing by avoiding objects that pop in and out.

We control the resulting v by introducing the *min-change* error metric $\delta(i,j)$ that may be used in place of $\Delta(i,j)$. Because we do not want dissimilar frames being included sequentially in v, we penalize $\delta(i,j)$ based on the dissimilarity between frames i and j:

$$\delta(i,j) = \sum_x \sum_y \left[\left(s_j^{xy} - s_i^{xy} \right)^2 \right] \tag{10}$$

Thus, $\delta(i,j)$ is the sum-of-squared difference between frames i and j. This calculation is reminiscent of the metric used for jump evaluation in Video Textures [Schödl et al. 2000].

As in Section 4.1, Equation 8 (substituted with $\delta(i,j)$) may be used to shape $\delta(i,j)$. Again, α and β are often near 1. If a noise offset is required, a per-jump γ error offset form is suggested:

$$\delta'(i,j) = (\beta \times MAX(0, \delta(i,j) - \gamma))^{\alpha} \tag{11}$$

4.3 Enforcing Uniformity

Up to this point, we have assumed that uniform sampling was undesireable. However, if samplings become very non-uniform, the selected frames tend to cluster together in close temporal proximity, resulting in playback of a segment rather than a summation. Thus, we introduce a modified error metric that can be used to bias the previous metrics towards uniform sampling.

As in Section 3, we determine the uniform discrete sampling interval T_d. We then attempt to include segments in v that are T_d apart and penalize others by how far they stray from T_d. This is

done with another error metric, $\Upsilon(i,j)$, this time with a normalized penalty based on dissimilarity from T_d:

$$\Upsilon(i,j) = \frac{j-i-T_d}{T_d} \tag{12}$$

This metric alone is useless, since it results in a uniform sampling. When used in conjunction with either the min-error $\Delta(i,j)$ or min-change $\delta(i,j)$ metrics (from Sections 4.1 and 4.2 respectively) it acts like a spring force pulling the solution towards uniformity. Thus, we present linearly-weighted combinations to accomplish this that we call the *combined uniform* error metrics:

$$\Delta(i,j)_{combined} = \lambda\Delta'(i,j) + (1-\lambda)\Upsilon(i,j) \tag{13}$$

$$\delta(i,j)_{combined} = \lambda\delta'(i,j) + (1-\lambda)\Upsilon(i,j) \tag{14}$$

Note, the shape variables α, β, and γ must act to normalize $\Delta'(i,j)$ and $\delta'(i,j)$ between 0 and 1 to be of relative scale to $\Upsilon(i,j)$.

4.4 Efficient Calculation

The slowest part of our technique is calculating $\Delta(i,j)$ (or $\delta(i,j)$) for every i,j pair. To accelerate the process, we can enforce the constraint that no two sequentially sampled frames contained in v may be more than q frames apart in s. In the array containing all $\Delta(i,j)$ values, this is equivalent to solving a band q values wide. This no longer guarantees an optimal solution, because time-lapse results cannot have a sampling interval larger than q. Care must be taken when choosing q because it impacts both the minimum and maximum jump lengths. For example, if two adjacent frames are chosen for v, then a future jump must become longer to cover the resulting gap. We typically choose a q 3 to 4 times larger than T_d.

This optimization, combined with other optimizations in [Bennett 2007], results in an overall time complexity of $\Theta(M \cdot N \cdot q)$.

5 Virtual Shutter

The virtual shutter computationally combines a sliding window of video-rate exposures from the input signal s to decrease aliasing, remove changing scene elements, or highlight motion paths.

The primary reason to use a post-process to create exposures is because it allows arbitrarily selection of any start and end time for each exposure. Thus, we are not constrained by the sampling interval being the maximum exposure time, as in a physical camera. In the virtual shutter, each video-rate input frame (which we assume is not saturated, not temporally aliased, and contains no motion blur) acts as a short, discrete time-step that we integrate into longer time-lapse exposures. Experimentally, these exposures were 1-10 ms, although 33 ms exposures were used at night. This approach frees us from the linear integration of CCDs or film, allowing per-frame integration by any weighting or non-linear function.

Furthermore, we have the capability to make exposures whose lengths adapt to the non-uniform sampling from Section 4. Since the time-lapse sampling results from a post-process, we cannot know the corresponding exposure lengths at capture time.

Each exposure interval is a function of the input sampling v of s. The user specifies how many sampling intervals the output should integrate over: Ψ. Values of $\Psi > 1$ imply overlaps between exposures. Specifically, an output frame z integrates the frames in s in the interval $v_{(z-\Psi)}$ to v_z (*i.e.*, a causal, non-symmetric process).

The virtual shutter can be considered a general purpose computational photography tool adapted for video. Computational photography has focused on processing a series of images as input and outputting a single frame. Alternatively, our virtual shutter slides a window of input frames through the video, creating an exposure for each v frame (Figure 4). Each output exposure is related to the image combinations in Image Stacks [Cohen et al. 2003]. However, we extend that work with our adaptive temporal window and by creating exposures that illustrate events as part of a video output.

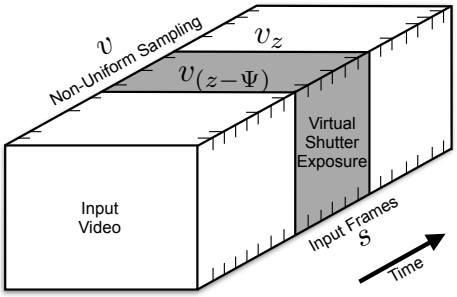

Figure 4: Illustration of the virtual shutter sliding a window of frames through the video that are combined into synthetic exposures. Each exposure ends at a sample v and extends Ψ indices in v back in time, allowing for adaptively-sized exposures.

Currently Supported Virtual Shutter Effects:

• **Maximum Virtual Shutter:** The maximum virtual shutter simulates film's non-linearity due to saturation. Nighttime time-lapse videos of car headlights are effective because the headlights saturate the film, creating bright, uniform streaks. We recreate a similar effect by choosing the maximum value at each pixel.

$$VS_z^{xy} = MAX_{f=v_{(z-\Psi)}}^{v_z} s_f^{xy} \tag{15}$$

• **Minimum Virtual Shutter:** The minimum virtual shutter alternatively chooses the darkest pixels within the exposure window. Although no photographic analog exists, this frequently removes bright foreground phenomena, such as fog or smoke.

$$VS_z^{xy} = MIN_{f=v_{(z-\Psi)}}^{v_z} s_f^{xy} \tag{16}$$

• **Median Virtual Shutter:** The median virtual shutter creates an exposure of the most representative pixels, even if those pixels never appeared simultaneously. This is another effect unachievable with a real camera. It is best used as a complement to the min-change non-uniform sampling (Section 4.2) to remove any residual popping artifacts that could not be sampled around.

$$VS_z^{xy} = MEDIAN_{f=v_{(z-\Psi)}}^{v_z} s_f^{xy} \tag{17}$$

• **Extended Exposure Virtual Shutter:** The extended exposure virtual shutter simulates holding the shutter open between exposures. This is accomplished with low-pass filtering to mimic the response of a camera along with normalization to avoid saturation.

$$VS_z^{xy} = \frac{1}{v_z - v_{(z-\Psi)} + 1} \sum_{f=v_{(z-\Psi)}}^{v_z} s_f^{xy} \tag{18}$$

The progression of time can be depicted by placing more weight on the most recent samples, indicating chronology.

$$VS_z^{xy} = \frac{1}{\sum \omega(z)} \sum_{f=v_{(z-\Psi)}}^{v_z} \omega(f) s_f^{xy} \tag{19}$$

We chose the $\omega(f)$ weighting to be an adaptive exponential falloff curve whose tail length matches the exposure window. We set μ to 30 and allow the user to configure the curve's falloff with ζ.

$$\omega(f) = \zeta^{\kappa}, \quad \kappa = \mu \cdot \frac{v_z - f}{v_z - v_{(z-\Psi)}}, \quad 0 < \zeta \leq 1, \quad 1 < \mu \tag{20}$$

• **Motion Tails Virtual Shutter:** The motion tails virtual shutter composites multiple frames together to simulate dense stroboscopic motion studies to illustrate paths of motion. The previous extended

Figure 5: Unprocessed input frames from each of the example videos discussed in Section 6. From left to right, a 13 second time-lapse of a candle burning with wax dripping, an 8 second sequence of a cookie baking, a 20 second summarization of the film "Reefer Madness", a 5 second time-lapse of people walking, a 6 second time-lapse of car headlights, a 10 second sequence of a car defrosting, a 7 second time-lapse of a street corner, and a 15 second time-lapse of a busy sidewalk during a class change.

exposure virtual shutter causes fast moving objects to fade into the background, as the background is seen more often than the motion. Now, we apply techniques from the "Image Stacks" [Cohen et al. 2003] matte filter that used the median as a background plate and image subtraction to extract foreground elements. These images were then composited using the "over" operation [Porter and Duff 1984] into the final image in temporal order, thus more recent frames' motions occlude older motions. We extend this idea by combining exposures in our sliding window of frames. If $\Psi > 1$, the resulting motion tails overlap. To better illustrate the direction of motion in our video output, we adjust the "over" α blending term using $\omega(f)$ (Eq. 20) to fade the older frames into the background.

6 Results

Our computational time-lapse system (Figure 2) is implemented as a series of applications that each perform a specific aspect of the overall process. The first application generates the $\Delta(i,j)$ min-error (Section 4.1) and $\delta(i,j)$ min-change (Section 4.2) metrics from the source video s (the $\Upsilon(i,j)$ uniformity metric from Section 4.3 is calculated on-the-fly). Here, we can opt to analyze specific spatial sub-regions of the video and choose to calculate at lower resolutions (often without noticeable degradation). These errors are then used by our non-uniform sampler application and associated GUI to perform dynamic programming optimization. Here, the user may specify the shape variables (α, β, γ, and λ) guided by an interactive visualization of their impact. The resulting sampling v is fed into our virtual shutter application, where the exposure effects of Section 5 are exported as a video.

For the remainder of this section, we describe how each sequence in our supplemental video (Fig. 5) was generated. The system settings and runtimes are detailed in Table 1 while the non-uniform samplings are shown in Figure 3. All videos are 720x480, except for "Reefer Madness" (320x240).

Non-Uniform Sampling Results:

• **Burning Candle** (*58 Minutes → 13 Seconds*)
This uniform 262x time-lapse of a candle burning misses the events of wax dripping and the candle breaking. To make a video with more of the dripping wax we primarily used the min-error metric on the candle's wax (masking the flame, which was not of interest), thus making the sampler approximate the wax activity.

• **Cookie Baking** (*9 Minutes → 8 Seconds*)
This uniform 65x time-lapse of a cookie baking is not temporally aliased, but the camera was slightly unsteady. Our stabilized output sampling used the min-change metric along with $\Upsilon(i,j)$ to identify similar frame pairs with a mostly-uniform (small λ) sampling.

• **"Reefer Madness"** (*68 Minutes → 20 Seconds*)
Playing the film uniformly at >200x results in incomprehensible scene flashes. Our non-uniform sampler, with the min-change metric, chose a few frames from longer scenes that appear similar, keeping the time-lapse within each scene longer for identification.

Virtual Shutter Results:

• **Building Front** (*34 Seconds → 5 Seconds*)
This short video of walking students shows the benefits of motion tails to convey motion, even with 7x uniform sampling. Low-pass filtering makes the students nearly disappear but motion tails of either $\Psi = 4$ or 8 make the motion contiguous and fluid (Fig. 7).

• **Nighttime Car Headlights** (*60 Seconds → 6 Seconds*)
A popular time-lapse effect images car headlights at night. We used the maximum virtual shutter to create exposures 3 and 6 times times longer than the maximum sampling interval allows (Fig. 6).

• **Car Defrosting** (*29 Minutes → 10 Seconds*)
This time-lapse of a car's windshield defrosting allows for several interpretations: minimizing the aliased motion (using the median virtual shutter) and also depicting all the motion with motion tails.

Combined Results:

• **Street Corner** (*12 Minutes → 7 Seconds*)
This 120x uniform time-lapse (Fig. 1) depicts a street corner with sporadic traffic, resulting in popping. With our min-error metric, we generated a non-uniform sampling that slows down when cars drive by. We further reduced aliasing with motion tails. Alternatively, we used the min-change metric (with a small λ), to sample the cloud motion without cars. Because some blowing leaves remained, we removed them with a short median virtual shutter.

• **Crowded Sidewalk** (*17 Minutes → 15 Seconds*)
A uniform 70x time-lapse shows a sidewalk during a class change that also exhibits popping. We used a min-error non-uniform sampling (and a tiny λ uniformity mix) to devote more samples to frames with increased activity (resulting in a busier sidewalk). We then added motion tails to show the students' paths (Fig. 7).

7 Future Work

There are many areas for further development of computational time-lapse video including several potential non-uniform sampling enhancements. The min-change metric $\delta(i,j)$ could be altered to compare neighborhoods of frames as opposed to single frames. As in "Video Textures" [Schödl et al. 2000], velocity would be preserved in addition to visual similarity. Also, repetitive motions could be cleverly resampled to give the illusion they were occur-

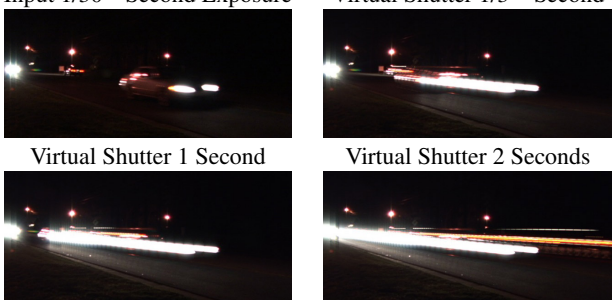

Figure 6: An input frame from the nighttime car headlights sequence and three extended exposures using the maximum virtual shutter to mimic film saturation. Although the longest sampling interval for a 10x speedup is 1/3rd of a second, we simulate longer exposures from video-rate footage.

	Duration (Frames)			Non-Uniform Error Metric				Uniformity	Virtual Shutter			Run Time
Title	**Input**	**Output**	q	**Sampling**	α	$\log \beta$	$\log \gamma$	λ	**Type**	Ψ	ζ	**hours:minutes**
Burning Candle	104849	400	600	Min-Error	1.09	-6.11	4.87	.05	-	-	-	7:20
Cookie Baking	15757	240	200	Min-Change	1.00	-5.24	0	.04	-	-	-	:05
"Reefer Madness"	40975	600	500	Min-Change	1.00	-7.08	5.89	-	-	-	-	:15
Building Front	1025	150	-	Uniform	-	-	-	-	Tails	4, 8	.96	:15
Car Headlights	3096	300	-	Uniform	-	-	-	-	Max.	6	-	:10
Car Defrosting	52544	300	-	Uniform	-	-	-	-	Median	10	-	:30
Car Defrosting	52544	300	-	Uniform	-	-	-	-	Tails	4	.93	:45
Street Corner - Cars	25483	210	500	Min-Error	1.00	0	0	-	Tails	3	.95	5:15
Street Corner - Clouds	25483	210	500	Min-Change	.65	-7.37	5.38	.15	Median	2	-	:45
Crowded Sidewalk	31128	450	500	Min-Error	1.00	-9.49	0	.01	Tails	4	.92	3:15

Table 1: Parameters used to generate the video results. These were selected manually by the user with our GUI. Cells marked as '-' indicate variables not used in the result. A value in the λ category indicates the combined uniformity metric $\Upsilon(i, j)$ was used along with the listed error metric. Runtimes reflect all processing (error calculation, sampling, and virtual shutter) on a 2 GHz Intel Core Duo. Most of the running time is spent on error metric calculations, which are a function of the metric, duration, q, and resolution. Also, note that the running-time of the min-error metric is much higher than the min-change metric, as it requires calculation at the all intermediate frames between i and j.

ring at video-rate speeds in the time-lapse result (thus, a desirable form of aliasing exploiting beat frequencies). Although we did not consider camera motion because time-lapse cameras are typically fixed, an error metric could be constructed to optimize for a constant camera velocity. A shortcoming of dynamic programming is that errors must be pairwise between i and j. Because objectives such as maximizing the smoothness of the derivative of intervals in v are not possible, other solution techniques should be considered.

The virtual shutter could be extended to support more varied combinations of frames, such as non-photorealistic rendering effects. Also, motion tails could be created with higher quality compositing and extended to support frame-to-frame registration for non-fixed cameras.

Finally, our methods could be extended into on-line versions that construct time-lapse videos on-the-fly. Although this would involve making sampling decisions without knowledge of the whole video, it would lower the disk storage requirements.

8 Conclusions

We have presented methods for creating computational time-lapse videos which provide superior sampling characteristics over traditional time-lapse methods. Based on user-specified characteristics, we generate time-lapse videos with non-linear filtering and non-uniform sampling, both not possible in traditional cameras. Our non-uniform sampling optimally chooses the frames of the time-lapse video using a general solution method that also supports alternate applications, such as summarization and stabilization. Our virtual shutter combines multiple video-rate exposures into longer exposures with reduced aliasing artifacts. Finally, we used these techniques to process a variety of typical and novel time-lapse videos.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D. H., AND COHEN, M. 2004. Interactive digital photomontage. *ACM Transactions on Graphics*, 294–302.

BENNETT, E. P., AND MCMILLAN, L. 2005. Video enhancement using per-pixel virtual exposures. *ACM Transactions on Graphics 24*, 3, 845–852.

BENNETT, E. P. 2007. *Computational Video Enhancement*. PhD thesis, The University of North Carolina at Chapel Hill.

BRAUN, M. 1995. *Picturing Time: The Work of Etienne-Jules Marey*. U. C. Press.

COHEN, M. F., COLBURN, R. A., AND DRUCKER, S. 2003. Image stacks. Tech. Rep. MSR-TR-2003-40, Microsoft Research.

DEMENTHON, D., KOBLA, V., AND DOERMANN, D. 1998. Video summarization by curve simplification. In *Proceedings of ACM Multimedia*, 211–218.

DIVAKARAN, A., PEKER, K. A., RADHAKRISHNAN, R., XIONG, Z., AND CABASSON, R. 2003. Video summarization using MPEG-7 motion activity and audio descriptors. Tech. Rep. TR-2003-34, Mitsubishi Electric Research Laboratory.

DOUGLAS, D., AND PEUCKER, T. 1973. Algorithm for the reduction of the number of points required to represent a line or its caricature. *The Canadian Cartographer 10*, 2, 112–122.

EDGERTON, H. E., AND KILLIAN, J. R. 1979. *Moments of Vision*. MIT Press.

HUA, X.-S., LU, L., AND ZHANG, H.-J. 2003. AVE: Automated home video editing. In *Proceedings of ACM Multimedia*, 490–497.

KINSMAN, E. 2006. The time-lapse photography FAQ. *www.sciencephotography.com*.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3, 277–286.

MUYBRIDGE, E. 1955. *The Human Figure In Motion*. Dover Publications.

PEREZ, J.-C., AND VIDAL, E. 1994. Optimum polygonal approximation of digitized curves. *Pattern Recognition Letters 15*, 743–750.

PEREZ, R., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Transactions on Graphics 22*, 3, 313–318.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. *Computer Graphics 19*, 3, 253–259.

RAV-ACHA, A., PRITCH, Y., AND PELEG, S. 2006. Making a long video short: Dynamic video synopsis. In *Proceedings of CVPR*, 435–441.

RIDDLE, P. N. 1979. *Time-Lapse Cinemicroscopy*. Academic Press, New York, NY.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proceedings of ACM SIGGRAPH*, 489–498.

SMITH, M. A., AND KANADE, T. 1997. Video skimming and characterization through the combination of image and language understanding techniques. Tech. Rep. CMU-CS-97-111, Carnegie Mellon University.

WILDEMUTH, B. M., MARCHIONINI, G., YANG, M., GEISLER, G., WILKENS, T., HUGHES, A., AND GRUSS, R. 2003. How fast is too fast? Evaluating fast forward surrogates for digital video. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, 221–230.

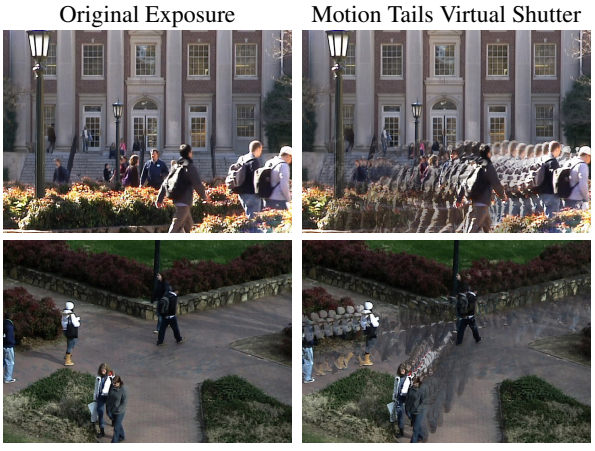

Figure 7: Two examples of using motion tails to depict dense motion paths between sampled time-lapse frames. The building front result (above) uses uniform sampling, while the crowded sidewalk (below) is non-uniformly sampled.