https://doi.org/10.2352/ISSN.2470-1173.2017.17.COIMG-430

Temporal super-resolution for time domain continuous imaging

Henry Dietz, Paul Eberhart, John Fike, Katie Long, and Clark Demaree;
Department of Electrical and Computer Engineering, University of Kentucky; Lexington, Kentucky

Abstract

Super-resolution (SR) image processing describes any technique by which the resolution of an imaging system is enhanced. Normally, the resolution being enhanced is spatial; images are processed to provide noise reduction, sub-pixel image localization, etc. Less often, it is used to enhance temporal properties – for example, to derive a higher framerate sequence from one or more lower framerate sequences. Time domain continuous imaging (TDCI) representations are inherently frameless, representing a time-varying scene as a compressed continuous waveform per pixel, but they still imply finite temporal resolution and accuracy. This paper explores computational methods by which the temporal resolution can be enhanced and temporal noise reduced using a TDCI representation.

Introduction

TDCI[1] representations offer a variety of benefits, some of which have been explored in publications at Electronic Imaging 2014, 2015, and 2016. For example, TDCI allows virtual exposures for images to be specified after capture, supporting arbitrary selection of the interval represented by a computationally extracted image while providing high dynamic range independent of virtual shutter speed. Beyond introducing the basic concepts, our earlier work centered on methods to capture TDCI streams directly or to synthesize them using conventional cameras. The current work centers on methods by which the temporal resolution can be improved beyond the shortest pixel integration time supported by the capture device. Unlike conventional video, TDCI streams can represent arbitrarily precise timing information. Thus, expensive computational enhancement of temporal quality can be performed and results encoded once, then used to cheaply render many images from the same TDCI stream – for example, rendering a movie at various framerates.

Although temporal interpolation between images is fairly common, our goal is not to simply create intermediate frames. Rather, the goal is to produce the highest possible temporal quality for each pixel value change – without imposing constraints that would require all pixels to change state simultaneously at frame edges. This paper explores various methods by which such temporal enhancement can be accomplished and how effective these methods can be.

What is known

Before discussing algorithms for super-resolution temporal interpolation, it is important to understand what a TDCI stream actually encodes. What are the empirically known data for a particular pixel?

There are some applications of imaging, primarily in the scientific and engineering domains, that are directly interested in measuring properties of photons. However, that is not what people care about when they look at a photograph. The underlying assumption of TDCI is that a photograph is intended to be a model of scene appearance. An image should look approximately like we would perceive the scene. Individual photons are merely the mechanism by which a camera samples that appearance; they are not part of the model per se.

For example, imagine that one is photographing a blue piece of paper. The paper's appearance is blue because the material of which it is made preferentially reflects a larger fraction of blue light than other colors. However, in some small time interval, there might only be a single red photon reflected. Conventional wisdom would argue that the paper is red during that time interval. In contrast, TDCI suggests that during that time period the paper is *probably* the same blue it was before and after that time interval, but the lack of sufficient photons to sample it makes us unable to prove our hypothesis. The more precisely one attempts to know *when* an object has a particular color and brightness, the lower the confidence and precision with which one can actually measure the color and brightness.

This implication of this is that all empirically-measured pixel values, even those made with theoretically perfect photon detectors, are noisy averages sampled over a period of time. The smaller the number of photons used to sample, the greater the uncertainty. Without control over the rate of photon arrival, the only way to sample enough photons to have high confidence in the pixel values is to sample over a relatively long interval. That additionally requires that the scene content itself not be changing during that time.

Most temporal super-resolution algorithms are really for frame-rate up-conversion; they focus on estimating how objects move within the scene between evenly-spaced frames[5], often also applying some filtering to reduce noise[6]. That emphasis is based on two assumptions:

1. The pixel values in a frame (an image within a temporal sequence of images) are correct; temporal interpolation must pass through these values at the recorded time
2. The majority of change between frames is due to changes in the scene appearance (primarily motion of scene elements relative to the camera); the scene is changing faster than the light by which it is sampled

The work in this paper makes neither of those assumptions, nor does it require that the scene is sampled a frame at a time nor at regular intervals. Pixel samples are noisy values within approximately knowable error bounds and most small, rapid, changes are due to noise rather than changes in scene appearance.

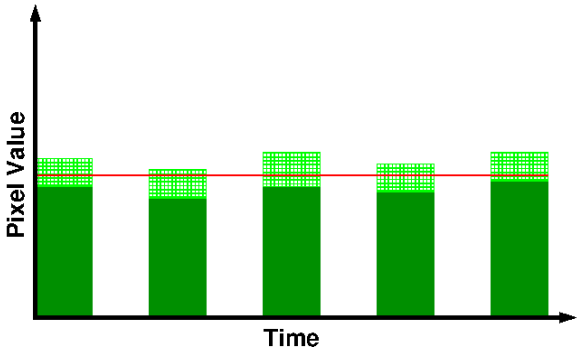

Figure 1. *Constant within error bounds is probably constant*

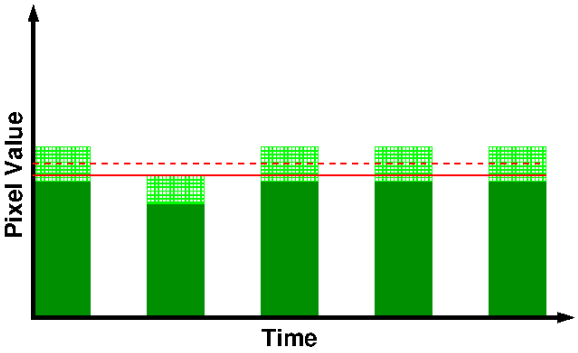

Figure 2. *Simple averaging can violate sample error bounds*

If objects within the scene move, they do not move far between temporal samples. The different assumptions made in this paper also reflect the idea that capture is made at higher temporal sampling frequencies or framerates – and **TDCI is intended primarily for image data with 1/240s or finer temporal resolution**. In sum, our goal is improving temporal and value accuracy of pixel waveforms to enable extraction of extremely high quality stills representing arbitrary time intervals.

Sample quality and error bounds

Before discussing methods for improving the quality of TDCI image data, it is important to note what is known about the pixel samples.

The time interval represented by each pixel sample is known very precisely, usually to an accuracy that is a small fraction of the shutter speed. Even samples timed in software (e.g., using CHDK[7] inside a Canon PowerShot) generally have timing known to within 0.001s. The problem is resolving pixel values at times within or between samples.

Error bounds on pixel samples are a much more complex thing to determine precisely, but we have several viable methodologies. The standard one used with TIK[3] is computed by analysis of a time sequence captured of a completely static scene using as close as possible to the same exposure parameters, ambient temperature, etc., that is used for the TDCI pixel data to be processed. TIK can perform the analysis of the test capture to produce a 256x256 map for each color channel in which the pixel values are scaled to 0..255 and the [x][y] entry reflects the probability that a pixel sampled with value y is subsequently sampled as the value x. These error maps are generated as PPM images, so they can be manipulated with image editing tools; for example, an ISO setting between ones for which error map images were experimentally determined can be approximated by weighted averaging of error maps from lower and higher ISOs.

The actual error model used here, and in TIK, is that of hard bounds on minimum and maximum values. These bounds are determined by setting a probability threshold which is then applied to the error map.

Purely temporal interpolation

The simplest methods for increasing temporal resolution examine the value of each pixel individually as it evolves over time. It would seem very straightforward to interpolate between the points on the one-dimensional trajectory of a pixel's values over time, but the pixel values read are not points:

- Each sampling of a pixel value represents an average measured over a time interval, not a reading at a point in time.
- Each sampling of a pixel value is subject to error due to noise and perhaps other corruptions, such as artifacts from lossy compression – as used for JPEG stills, MPEG video, etc.

It is easy for a simple interpolation process to magnify these errors, actually synthesizing temporal noise. The goal in purely temporal interpolation is to maximize the probability that the interpolated values reflect what the true pixel values would have been at each point in time.

Variation within error bounds

Most of the area of most scenes does not change appearance from one frame to the next. This should be by far the most common case for the evolution of each pixel's sampled value over time... but it isn't. In fact, it is very rare that the value is identical from one sample to the next. Noise and other corruptions of the data cause small, largely random, variations in the pixel value sampled over time.

A simple example of this is depicted in Figure 1. Each of the green blocks represents a pixel sample, with width equal to the exposure integration time (shutter speed) and height equal to the average value of the pixel in that interval. The partially-shaded region at the top of each green block represents the error bounds on that value. The red line shows a highly credible interpolated value – a constant value that passes within the error bounds for each sample.

The fact that the red line is constrained by multiple slightly-different bounds essentially gives it higher accuracy than any of the individual readings could afford. This is the basic principle behind image stacking[2, 4] as it is commonly used in astro photography. Corresponding pixel value samples from a time sequence of aligned images of the exact same scene are averaged, often dramatically improving both dynamic range and signal-to-noise ratio (SNR).

Our initial implementation of interpolation in TIK[3] recognizes when temporally-adjacent pixel samples have overlapping

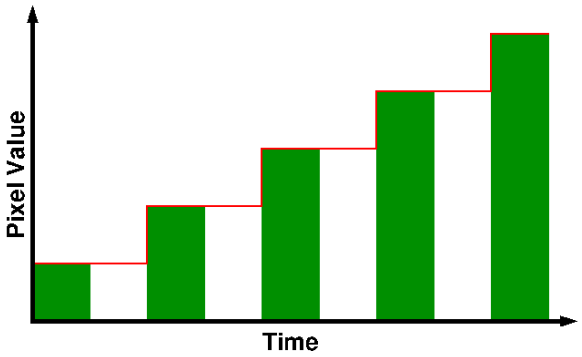

Figure 3. *How conventional video handles slopes*

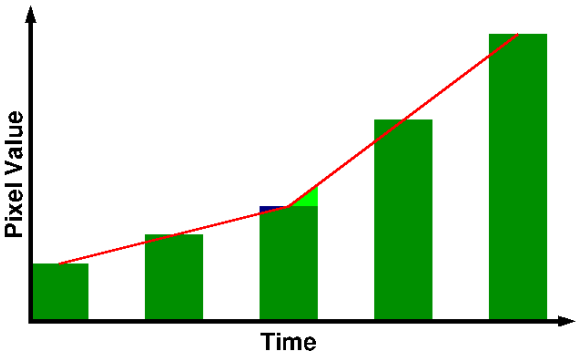

Figure 5. *Linear interpolation between sample centers fails*

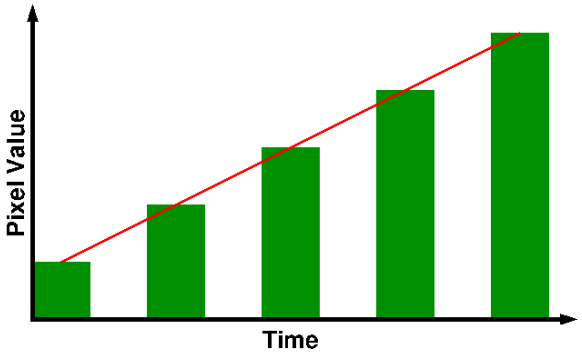

Figure 4. *Linear interpolation between sample centers*

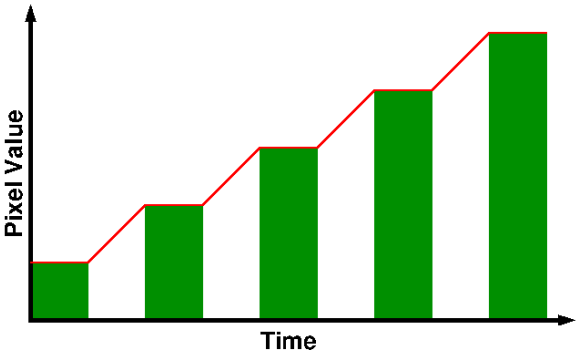

Figure 6. *Simple linear interpolation between sample intervals*

error bounds and combines those samples. The combining can be as simple as averaging the reported values for all samples, but simple averaging can result in pixel values that land outside the error bounds for some samples. Figure 2 shows a case in which the average, shown with a dashed red line, would fall outside the error bounds for the second sample.

Assuming that the error bounds are in fact correct, the constant value selected to cover all temporally-adjacent samples with overlapping bounds must reside within the intersection of all the bounds. On that basis, we argue for the following procedure, which produces the solid red line shown in Figure 2:

1. Determine the average of the pixel sample values,
   $avg = (\sum_{n=1}^{N} sample_n)/N$
2. Determine the lower bound on the intersection,
   $min = maximum(min_1, min_2, ...min_N)$
3. Determine the upper bound on the intersection,
   $max = minimum(max_1, max_2, ...max_N)$
4. Find value in bounds nearest to average,

$$value = \begin{cases} min & \text{if } avg < min; \\ max & \text{if } avg > max; \\ avg & \end{cases}$$

In fact, this concept of the average staying within bounds also can be used to detect when a sequence of stackable values actually hides a slowly-changing scene. Slow dimming or brightening of the scene can produce overlapping bounds that skew higher or lower over time. To avoid interpreting such a shallow, and noisy, slope as a constant, one could simply end the sequence as producing a constant when the average value first hits the intersection minimum or maximum. In such a case, only the third through fifth samples of Figure 2 would be treated as having a constant true value.

Slopes

Interpolation of an essentially constant value is not really improving its temporal resolution. Temporal super-resolution really happens only when sample values are changing by significant amounts.

Let us begin by considering interpolation along samples following a simple slope, as shown in Figure 3. In this example, we make the simplifying assumption that each of the sample values is fully precise and accurate; the min and max error bounds are the sample value.

The normal handling of video sequences is that, although the shutter speed may be relatively fast, each pixel sample is treated as though it represented the true pixel value for the entire period up to initiating capture of the next frame. For example, if the width of a sample in Figure 3 is 1/60s, the framerate is 1/30s and the second 1/60s of each 1/30s interval is assumed to be identical to the first half. In cinematography, the ratio between the length of the exposure interval and the time period per frame is commonly known as the shutter angle and expressed as $angle = 360^o \times shutter/period$. Thus, the example would be said to have a 180^o shutter angle.

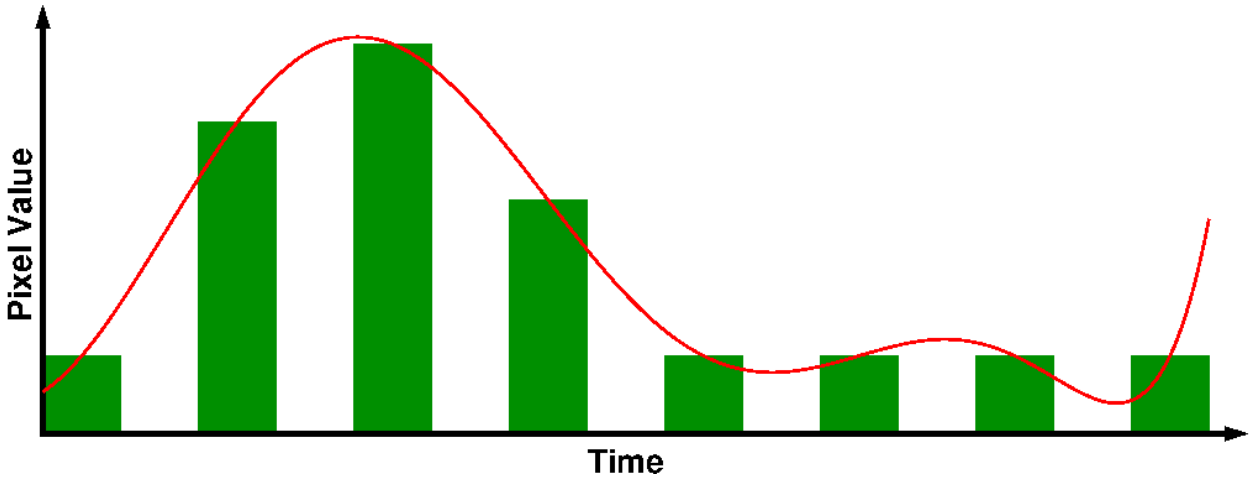

Figure 7. *Lagrange (and other polynomial) interpolation has trouble lying flat*

The traditional cinematographic handling of a slope results in very abrupt temporal steps with temporal resolution restricted to the period of the framerate. An easy, probably more accurate, way to handle this is to simply draw a line from the center of each sample to the next, as shown in Figure 4. For a truly constant rate of change, this has some very desirable properties.

However, consider what happens when the slope changes, as shown in Figure 5. Linear interpolation between the centers still appears perfectly reasonable, except for one detail: if we were to compute a value for the same time interval represented by the third sample, we would get a different value! To be precise, the light-green area included above the sample value is greater than the dark blue area excluded, so the value computed would be somewhat higher than the actual sample. Even if fairly generous error bounds were permitted, the value computed could still be shifted outside of those bounds.

We suggest that a basic principle in super-resolution rendering should be that **sampling the interpolated function in any interval that corresponds directly to an original sample should never result in a value outside the error bounds of the original sample**. Linear interpolation based on sample center points in general will violate this principle whenever the slope changes.

With the priority that original sample error bounds not be violated, the easiest solution is to interpolate only between sample periods. Each sample endpoint is connected to the start of the next with a straight line. This very simple approach is illustrated in Figure 6, and it is the method currently used in in TIK[3]. Empirically, performance is quite good using this method – transitions are smoother than one would expect. Why? Normally, virtual exposures are being made with integration times that are not much shorter than those used for the original samples. It is common that a virtual sample will misalign with the edge of an original sample, thus integrating a portion of the original sample value and the sloped transition; one doesn't see a sudden transition. Further, the values centered between samples are precisely the average of those samples.

There is an additional benefit to this type of linear interpolation between samples. If we have determined that a particular sequence of samples represents a constant, we can **replace the entire sequence of samples with a single virtual sample spanning the full temporal interval** and having relatively tight error bounds derived by intersection. In fact, this is the primary method by which the basic TIK implementation compresses TDCI data streams. This type of substitution would greatly magnify errors if used with the point-based linear interpolation.

Curves

Although linear interpolation between sample intervals is quite effective, there are still abrupt changes in slope of the pixel value waveform over time. Interpolation is a very heavily-studied field, and there are many methods for generating a smooth curve (possibly even with smooth derivatives) to represent a data set. Most methods center on finding piecewise-polynomial curves which respond to a set of control points in ways dictated by a set of knotts and basis functions. Various types of Splines including Non-Uniform Rational Basis Splines (NURBS), Bezier curves, and Lagrange interpolaters are among the more common methods.

The unfortunate problem is that fitting a smooth curve to data points is not the task at hand. Each sample of a pixel's value merely defines an average over an interval – not a value at a point. Adding to that distinction is the fact that even the average values are not known precisely, but as estimates within error bounds.

Figure 7 shows a simple data set and the result of Lagrange interpolation. The control points used were the centers of the sample intervals and the sample values. The C-coded Lagrange interpolator we created for this actually has the additional unusual feature that it can iteratively adjust the control points within the error bounds to try to ensure that virtual exposures for each of the original sample intervals would result in a value within the original error bounds, but in this particular run the error bounds were zero. The key point is that the interpolator handles the constant time range very badly, introducing noise where there was none. Similar issues appeared with the other curve interpolators tested – polynomials tend not to lie perfectly flat.

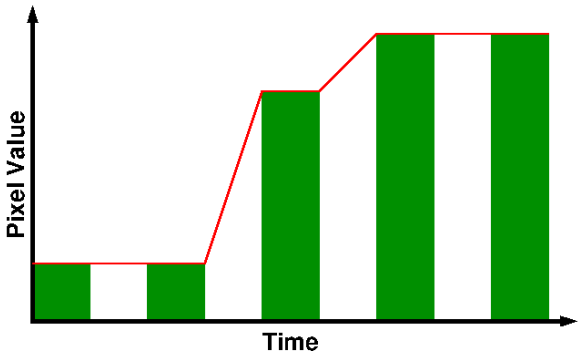

Figure 8. *A linear approximation to a smooth interpretation*

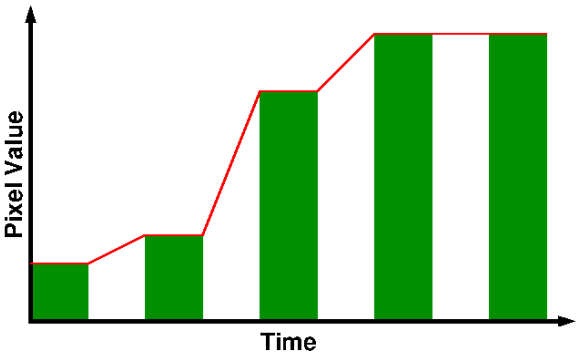

Figure 10. *Multiple transitional samples imply a smooth interpretation*

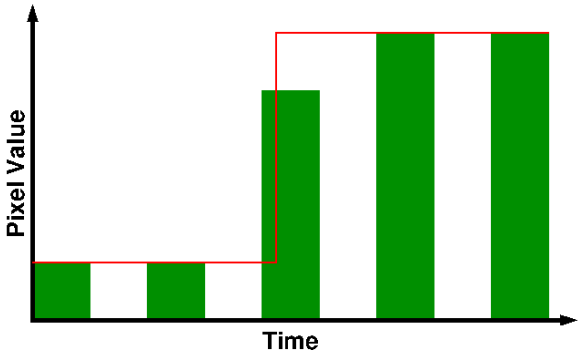

Figure 9. *Temporal super-resolution localization of an edge*

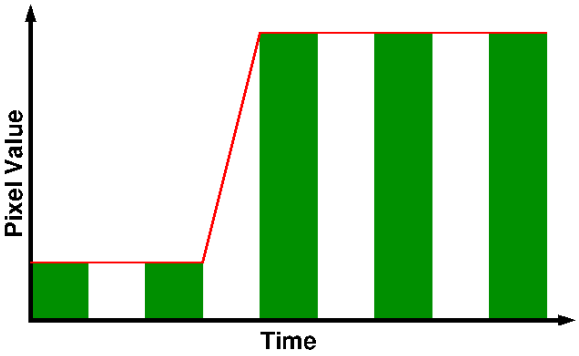

Figure 11. *Edge between samples cannot be localized further*

Spatio-temporal interpolation

All the above techniques assumed that the pixel value waveforms were essentially smooth. However, relative motion of the scene can change the object that a pixel is sampling, and that can cause strongly discontinuous transitions. It is in recognizing these discontinuities that spatial information – values of nearby pixels – play a role.

Temporal edge localization

Consider the transition shown in Figure 8. Using TIK's linear interpolation between sample intervals, the transition is treated as a relatively smooth three-segment linear approximation to a nonlinear curve brightening the pixel.

However, the first two samples establish a very consistent constant level. Similarly, the fourth and fifth samples seem to define a new constant level for the pixel. Suppose that we somehow know that in fact the dark level was a person's dark jacket as they walked through a doorway and the brighter level was the sky seen through the doorway after the person had passed through. In that case, there really isn't a smooth transition; the pixel goes directly from seeing the dark coat to seeing the bright sky. If that is the case, we can deduce that the in-between value of the third sample must have come from summing partial exposures to the coat and sky. The fact that the third sample's value is 3/4 of the way between the constant levels thus implies that 1/4 of the sample time was seeing the coat and 3/4 were seeing sky – suggesting the sharp edge shown in Figure 9.

The catch is that it isn't reasonable for our interpolator to recognize the coat and sky, so how can we obtain information that will disambiguate between the interpretations in Figures 8 and 9?

The first problem to solve is recognition of the constant levels. Fortunately, that is trivially accomplished using the error model discussed earlier.

The second problem has to do with detecting continuity of motion. The reasoning is that any roughly continuous motion of a sharp-edged object should only cause a specific pixel to transition from sampling one value to the other at most once. That single transition could happen between samples, or it could happen within a sample interval. If it happens between, it will not be seen. However, as shown in Figure 11, the error in the linear interpolation between sample intervals is already known to be less than half the temporal gap between samples.

The third and final problem is knowing that there is indeed a sharp edge between the two constant levels. This is where looking at spatially neighboring pixel values becomes useful. If there is such an edge, then one or more pixels on opposite sides of the current pixel should be detecting each of the two constant levels at appropriate times (an overlapping time interval). Basically, each side of the edge has to come from some neighbor and end in another. Failing this constraint suggests a scene structure spatially sampled below Nyquist, in which case creating a smoothed structure is preferable to synthesizing a sharp edge.

Temporal synchronization

The time at which a transition occurs should be highly correlated for neighboring pixels. Where this correlation occurs, more precise timing may be derived by interpolating the time from neighboring pixels.

Suppose that a particular pixel has a non-sharp (interpolated) transition between samples, such as appears in Figure 11. If we imagine that each sample in that figure took 8 "ticks" of time, then the ambiguity is that the level transition came between T=24 and T=32. Now suppose that one of the adjacent pixels had a super-resolution transition at time T=36, as shown in Figure 9. Further, suppose that another neighboring pixel, on the opposite side of the non-sharp transition pixel, also had a super-resolution transition, but at time T=22. The average of the two sharp transition times is (36+22)/2, or T=29. Since T=29 is in the interval 24..32, it is reasonable to assume that the previously non-sharp transition should be corrected to a sharp transition at T=29.

In essence, the idea is that **times of similar super-resolution transition events in neighboring pixels can be spatially interpolated** in nearly the same way that color information is interpolated to demosaic Bayer-filtered pixels. With some additional processing logic, even non-sharp transition events can be used in this way to develop time-value constraints on neighboring pixels: the transition timing can be sharpened to the interpolated temporal intersection of the possible transition time intervals. It is also possible to extend the concept of "neighbors" beyond immediate neighbors to a small region around the pixel in question.

Once a pixel's waveform has had a portion of its timing enhanced in this way, that enhanced information also may be propagated to improve timing of other neighbors.

Timing of pixel integration intervals

Although all the figures in this paper have shown pixel samples evenly spaced in time, there is no such constraint on TDCI data. Neither is it necessary that the input nor output be organized as frames in which all pixels have identical integration intervals. In fact, skewing of light integration periods across a sensor is highly desirable, because it dramatically improves the probability that waveforms of neighboring pixels can be used to refine timing information.

There are a variety of methods that can be used to control the integration intervals for pixels, some of which can enhance the effectiveness of temporal super-resolution processing:

- **Native TDCI capture**: as described in our earlier work on frameless TDCI capture[1], there are various ways that an imaging sensor can be constructed to either have a fully independent, or deliberately pattern skewed, sampling interval for each pixel. Unfortunately, it is not clear that any currently available camera sensor is capable of directly implementing this. We have created various prototype cameras that approximate this behavior, such as the FourSee multi-camera[8], which deliberately temporally skews exposures from multiple cameras sharing a single lens and viewpoint.
- **Leaf shutter**: an aperture-like iris opens and then closes. As the aperture is opening, the lens passes through a variety of effective f/numbers, eventually opening enough so that the aperture is determined by the aperture in the lens. The same thing happens in reverse to end the exposure. Thus, over the exposure interval the aperture varies somewhat, making light sensitivity a function of time. The effect is usually negligible, as is the associated change in depth of field during the exposure interval.
- **Focal plane shutter**: in most interchangeable-lens cameras, a first curtain sweeps across the sensor to begin exposure and then a second follows it to recover the sensor ending exposure. Older single-lens reflex (SLR) cameras often had focal plane curtains moving horizontally, but now traveling the shorter vertical distance is more common. In either case, the exposure interval is a function of respectively the pixel X or Y coordinates. Although focal plane shutter speeds of 1/4000s (250us) are not uncommon, the curtain traversal time is usually closer to 1/200s (5000us). Thus, at a shutter speed of 1/4000s, the last pixels of the sensor are exposed approximately 20 exposure intervals later than the first. The fractional skew is less at low shutter speeds; a 1s exposure would be temporally skewed only 0.5% from one edge of the sensor to the other.
- **Electronic rolling shutter**: in most webcams and video cameras, there is no mechanical shutter. Instead, the system takes advantage of the fact that pixels are addressable elements. All the pixels are scanned to begin or end exposure, typically in a raster order. The temporal effect is very similar to that of a focal plane shutter, except the scan is often slower and, although one axis suffers more delay than the other, when pixels are sampled usually is a function of both X and Y coordinates.
- **Global electronic shutter**: although it is rare at this writing, some sensors have mechanisms that allow all pixels to start or stop collecting change from photons simultaneously. In such a system, all pixels truly sample during the exact same time interval. Of course, this is arguably the worst case when attempting to perform temporal super-resolution enhancement.

The temporal skew caused by focal plane or electronic rolling shutters is easily measured to a very high accuracy[9], but is often considered problematic. However, it means that nearby pixels are time shifted by a small and precisely knowable amount. Thus, an edge that falls between samples for a particular pixel might well fall within a sample for one of its neighbors. Similarly, the temporally skewed samples from similar events seen by a group of pixels can be combined to make a temporally-denser set of samples for all pixels. By recognizing patterns and temporally aligning the waveforms from adjacent pixels, the more-precise timing of an event for one pixel may be used to tune the record of when the corresponding event happened for another pixel. In summary, temporal sampling skew simply makes temporal synchronization more effective.

Conclusion

Temporal super-resolution using various forms of frame-oriented data has been done many times before. The goal is generally to insert frames between existing ones, thus up-converting the frame rate. However, TDCI is very different. Because it is frameless, the goal is not synthesizing equally-spaced frames, but enhancing the temporal and value accuracy of per-pixel waveforms. Even tiny adjustments of when value transitions occur can significantly improve virtual exposures rendered by integrating pixel waveforms over an arbitrary time interval.

The work reported in this paper is still very preliminary; we do not yet have enough data to make definitive statements about the effectiveness of the various methods discussed. However, several key ideas are noteworthy:

- At high sampling rates, value error (noise) is more significant than large-scale object motion
- Any changes made to the TDCI pixel waveforms should respect the error bounds on the relevant pixel samples
- Polynomial interpolation functions do not directly apply; the problem is not interpolating between known points because pixel samples really are bounded estimates of average values over short time periods
- Temporal skew in pixel sampling can be beneficial

Some of the methods discussed here are already implemented in TIK, and various sample images are included in the 2017 Electronic Imaging paper describing TIK[3]. Ongoing work centers on improving that tool.

Acknowledgments

This work is supported in part under NSF Award #1422811, *CSR: Small: Computational Support for Time Domain Continuous Imaging.*

References

[1] Henry Gordon Dietz, Frameless, time domain continuous image capture, *Proc. SPIE 9022, Image Sensors and Imaging Systems 2014, 902207 (March 4, 2014); doi:10.1117/12.2040016.* (2014).

[2] Richard L. White1, David J. Helfand, Robert H. Becker, Eilat Glikman, and Wim de Vries, Signals from the Noise: Image Stacking for Quasars in the FIRST Survey, *The Astrophysical Journal, Volume 654, Number 1; http://stacks.iop.org/0004-637X/654/i=1/a=99* (2007).

[3] Henry Dietz, Paul Eberhart, John Fike, Katie Long, Clark Demaree, and Jong Wu, TIK: a time domain continuous imaging testbed using conventional still images and video, accepted to appear in *IS&T Electronic Imaging, Image Sensors and Imaging Systems 2017, pp. 1-8 (February 1, 2017)* (2017).

[4] Deep Sky Stacker, *http://deepskystacker.free.fr/* (accessed November 26, 2016).

[5] Tsung-Han Tsai, An-Ting Shi, and Ko-Ting Huang, Accurate Frame Rate Up-Conversion for Advanced Visual Quality, *IEEE transactions on broadcasting , 2016, Vol.62(2), p.426-435*, (2016).

[6] J. C. Brailean, R. P. Kleihorst, S. Efstratiadis, A. K. Katsaggelos, and R. L. Lagendijk, Noise reduction filters for dynamic image sequences: a review, *Proceedings of the IEEE, vol. 83, no. 9, pp.1272-1292, doi:10.1109/5.406412*, (1995).

[7] Canon Hack Development Kit (CHDK), *http://chdk.wikia.com/wiki/CHDK* (accessed November 26, 2016).

[8] Henry Gordon Dietz, Zachary Snyder, John Fike, and Pablo Quevedo, Scene appearance change as framerate approaches infinity, *Electronic Imaging, Digital Photography and Mobile Imaging XII, pp. 1-7 (February 14, 2016);* (2016).

[9] Helmut Dersch, Subframe Video Post-Synchronization, *http://webuser.fh-furtwangen.de/d̃ersch/sync/sync_1.pdf* (November 24, 2016).

Author Biography

Henry (Hank) Dietz is a Professor in the Electrical and Computer Engineering Department of the University of Kentucky. He and the student co-authors of this paper, Paul Eberhart, John Fike, Katie Long, and Clark Demaree, have been working to make Time Domain Continuous Image capture and processing practical. See `Aggregate.Org` *for more information about their research on TDCI and a wide range of computer engineering topics.*