A Multiscale Algorithm for Reconstructing Videos from Streaming Compressive Measurements

Jae Young Park and Michael B. Wakin

Abstract

We propose a multiscale, iterative algorithm for reconstructing video signals from streaming compressive measurements. Our algorithm is based on the observation that, at the imaging sensor, many videos should have limited temporal bandwidth due to the spatial lowpass filtering that is inherent in typical imaging systems. Under modest assumptions about the motion of objects in the scene, this spatial filtering prevents the temporal complexity of the video from being arbitrarily high. Thus, even though streaming measurement systems may measure a video thousands of times per second, we propose an algorithm that only involves reconstructing a much lower rate stream of "anchor frames." Our analysis of the temporal complexity of videos reveals an interesting tradeoff between the spatial resolution of the camera, the speed of any moving objects, and the temporal bandwidth of the video. We leverage this tradeoff in proposing a multiscale reconstruction algorithm that alternates between video reconstruction and motion estimation as it produces finer resolution estimates of the video.

I. Introduction

A. Motivation and Overview

The emerging theory of Compressive Sensing (CS) has inspired a number of efficient new designs for signal acquisition in general and imaging in particular. Architectures such as the "single-pixel camera" [2, 3] provide a promising proof-of-concept that still images can be acquired using small numbers of randomized measurements. Despite the apparently incomplete data collected by such devices, reconstruction of the signal can be accomplished by employing a sparse model specifying, for example, that a high-dimensional image may have only a small number of significant coefficients when expanded in the two-dimensional (2D) wavelet domain [4].

There are numerous applications where it could be helpful to extend the CS imaging framework beyond still images to incorporate video. Standard video capture systems require a complete set of samples to be obtained for each frame, at which point a compression algorithm may be applied to exploit spatial and temporal redundancy. In

JYP is with the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. MBW is with the Department of Electrical Engineering and Computer Science at the Colorado School of Mines, Golden. Email: jaeypark@umich.edu, mwakin@mines.edu. This research was partially supported by DARPA Grant HR0011-08-1-0078, AFOSR Grant FA9550-09-1-0465, and ONR Grant N00014-09-1-1162. A preliminary version of the algorithm in this paper was originally presented at the 2009 Picture Coding Symposium [1].

some applications, such as imaging at non-visible (e.g., infrared) wavelengths, it may be difficult or expensive to obtain these raw samples. In other applications, it could be computationally challenging to implement a state-of-the-art video compression algorithm at the sensor. We argue that these burdens may be reduced by using compressive imaging hardware where random measurements are collected independently from each frame in the video and no additional compression protocol is needed.

From a hardware perspective, it is not difficult to envision extending standard compressive imaging architectures to acquire compressive measurements of a video. For example, the single-pixel camera takes random measurements serially in time. Each measurement corresponds to a random linear function of the image on the focal plane at the instant that measurement is collected. When the single-pixel camera is photographing a fixed scene, each measurement corresponds to the same image. When the single-pixel camera is photographing a scene containing motion, each measurement will correspond to a different "frame" of the video. The single-pixel camera is capable of taking many thousands of measurements per second. While we will continue to use the single-pixel camera as a specific example in this paper, other compressive imaging architectures (such as a CMOS-based transform imager [5] and a coded-aperture imager [6]) could be similarly used to acquire streaming measurements of a video.

From a data processing perspective, though, there are two major reasons why implementing a CS video system may be significantly more difficult than implementing a CS imaging system:

- **Challenge 1:** The complexity of a CS reconstruction algorithm is dependent on the number of unknowns that must be recovered from the compressive samples. The sheer volume of data in a raw video signal makes CS reconstruction a formidable task. As we explain in Section III, this problem is only exacerbated in compressive video systems that collect streaming measurements, where the number of measured frames can be in the thousands per second. Naively reconstructing all of the pixels in all of these frames could literally involve solving for billions of unknowns every second.
- **Challenge 2:** Reconstructing a signal from compressive measurements requires an efficient sparsifying transform and a corresponding algorithm that can promote this sparsity in the reconstructed signal. In the long literature of standard video compression [7] (not video CS), a variety of methods have been proposed to exploit spatial and temporal redundancies. One common approach combines motion compensation and estimation algorithms [8] with image compression techniques; for example, given a set of vectors describing the motion in the video, the LIMAT framework [9] yields a motion-compensated wavelet transform (across the temporal and spatial dimensions) intended to provide a sparse representation of the video. While some of these central ideas can be absorbed into the CS framework, there is an important challenge that we must address. Unlike the standard video compression problem where the frames of the video are explicitly available to perform motion estimation, in CS

only random measurements of the underlying video are available. We are faced with a chicken-or-egg problem [1]: Given the video frames, we could estimate the motion; but given the motion we could better estimate the frames themselves.

In this paper, we offer suggestions for confronting both of these challenges. We begin in Section II with a short discussion concerning the temporal bandwidth of video signals. We argue analytically that, at the imaging sensor, many videos should have limited temporal bandwidth due to the spatial lowpass filtering that is inherent in typical imaging systems. Under modest assumptions about the motion of objects in the scene, this spatial filtering prevents the temporal complexity of the video from being arbitrarily high.

We then explain in Section III how this limited temporal complexity can be exploited in addressing Challenge 1 above. Following standard arguments in sampling theory, we note that under various interpolation kernels, a stream of high-rate video frames (such as those measured by a single-pixel camera) can be represented as a linear combination of a low-rate (e.g., Nyquist-rate) "anchor" set of sampled video frames. We then explain how the CS video problem can be reformulated by setting up a system of linear equations that relate the compressive measurements to the underlying degrees of freedom of the video (specifically, the anchor frames). This significantly reduces the number of unknowns that must be solved for. As we demonstrate, our use of interpolation kernels for reducing the burden of processing streaming measurements can also be much more effective than the traditional technique of partitioning the measurements into short groups and assuming that all measurements within a group come from the same frame. Such raw aggregation of measurements—which actually corresponds to using a rectangular interpolation kernel in our formulation—can introduce significant interpolation error and degrade the reconstruction quality.

Our analysis of the temporal complexity of videos reveals an interesting tradeoff between the spatial resolution of the camera, the speed of any moving objects, and the temporal bandwidth of the video. In Section IV, we explain how this tradeoff can be leveraged to address Challenge 2 above. We propose a novel, multiscale algorithm for reconstructing video signals from compressive measurements. Our algorithm begins by reconstructing a coarse-scale approximation to the video, having low spatial and temporal resolution. From this, we obtain a crude estimate of the motion vectors in the video, and we then use these motion vectors to define a sparsifying transform that enables reconstruction of the next-finer scale approximation to the video. Our representation framework is built around the LIMAT [9] method for standard video compression, in which motion compensation is used to improve sparsity in the three-dimensional (3D) wavelet domain. We solve the chicken-or-egg problem by alternating between motion estimation and video reconstruction, proceeding to higher and higher spatial and temporal resolutions. At each scale, we employ the anchor frames described above to manage the complexity of the reconstruction process.

We conclude in Section V by describing the differences between our work and several other important ones in the literature. We also present simulations that demonstrate the performance of our algorithm. We stress that to some degree, the algorithm we present in this paper is a proof-of-concept inspired by our temporal bandwidth analysis. We see our work as an addition to—not a replacement for—the nascent CS video literature, and we believe that the ideas we expound (such as using anchor frames to reduce the complexity of reconstruction) could be combined with other existing ideas in the literature.

II. On the Temporal Bandwidth of Video Signals

A. Setup

1) Signal model and objectives: For the sake of simplicity, we begin by considering "videos" that have just one spatial dimension; we extend this to videos with two spatial dimensions in Section II-D. We also begin by considering continuous-time, continuous-space videos; our analysis will reveal the implications of sampling these videos. We use the variable $t \in \mathbb{R}$ to index time (which we measure in seconds), and we use the variable $x \in \mathbb{R}$ to index spatial position on the focal plane. For convenience, we measure x in an arbitrary real-valued unit we call "pix"; this unit is intended to symbolize what might be the typical pixel size in a subsequent discretization of this video. One could easily replace pix with micrometers or any other arbitrary unit of distance.

We consider videos belonging to a simple but representative translational model. Let $g(x)$ denote a 1D function of space (think of this as a continuous-space "still image"), and consider a continuous-space, continuous-time video $f(x,t)$ in which each "frame" of the video merely consists of a shifted version of this prototype frame. More formally, suppose that

$$f(x,t) = g(x - h(t)),$$

where $h(t)$ is some function that controls how much (in pix) the prototype frame is shifted in the focal plane at each time step. Because we have an interest in video imaging systems with high temporal sampling rates, our purpose is to characterize the temporal bandwidth of the video $f(x,t)$.

In particular, *we suggest that $f(x,t)$ could have limited temporal bandwidth in plausible scenarios where the prototype frame $g(x)$ and translation signal $h(t)$ have limited complexity*. For example, in a physical imaging system, one may envision $f(x,t)$ as the video that exists at the focal plane prior to being sampled by the imaging sensor. It is reasonable to expect that, due to optical blurring and due to the implicit filtering that occurs from the spatial extent of each light integrator, the prototype frame $g(x)$ will have limited spatial bandwidth. Similarly, if the camera motion is constrained or due to the physics governing the movement of objects in the scene, one might expect that the translation signal $h(t)$ will have limited slope and/or limited temporal bandwidth. In the sections that follow, we explain how such scenarios can allow us to bound the approximate temporal bandwidth of $f(x,t)$.

2) Fourier setup: Let $F(\omega_x, \omega_t)$ denote the 2D Fourier transform of $f(x,t)$, and let $G(\omega_x)$ denote the 1D Fourier transform of $g(x)$; in terms of units, ω_x is measured in rad/pix, and ω_t is measured in rad/s. From the separability property of the 2D Fourier transform and the shift property of the 1D Fourier transform, it follows that

$$F(\omega_x, \omega_t) = G(\omega_x) \cdot L(\omega_x, \omega_t),$$

where

$$L(\omega_x, \omega_t) := \mathcal{F}_t\{e^{-j\omega_x h(t)}\}(\omega_x, \omega_t) \tag{1}$$

and $\mathcal{F}_t\{\cdot\}$ denotes an operator that performs a 1D Fourier transform in the temporal direction. That is, for fixed ω_x, $L(\omega_x, \omega_t)$ equals the 1D Fourier transform of $e^{-j\omega_x h(t)}$ with respect to time, evaluated at the frequency ω_t.

B. Temporal Bandwidth Analysis

The appearance of the $h(t)$ term within an exponent in (1) can complicate the task of characterizing the bandwidth of $f(x,t)$. However, by imposing certain assumptions on $h(t)$, this analysis can become tractable.

1) Constant velocity model for $h(t)$: As a starting example, we briefly discuss a "constant velocity" model for $h(t)$ that is commonly seen in textbook discussions of video bandwidth (see, e.g., [10]). We assume that $h(t) = \Gamma t$ for some constant Γ (having units of pix/s). In this case we have $L(\omega_x, \omega_t) = \delta(\omega_t + \omega_x \Gamma)$, and so $F(\omega_x, \omega_t) = G(\omega_x) \cdot \delta(\omega_t + \omega_x \Gamma)$, which corresponds to a diagonal line in the 2D Fourier plane with slope ($\Delta\omega_t$ over $\Delta\omega_x$) that depends linearly on Γ.

To see the implications of this in terms of bandwidth, suppose that $G(\omega_x)$ is bandlimited (or essentially bandlimited) to some range of frequencies $\omega_x \in [-\Omega_x, \Omega_x]$ rad/pix. (Again, even if the moving object has sharp edges, $G(\omega_x)$ may be bandlimited due to blurring in the imaging system.) In this case, it follows that $F(\omega_x, \omega_t)$ must be bandlimited (or essentially bandlimited) to the range of frequencies $(\omega_x, \omega_t) \in [-\Omega_x, \Omega_x] \times [-\Gamma\Omega_x, \Gamma\Omega_x]$. In other words, the temporal bandwidth of the video is no greater than $\Gamma\Omega_x$ rad/s.

2) Bounded velocity model for $h(t)$: We now consider a more robust "bounded velocity" model for $h(t)$; we note that similar mathematics have appeared in the analysis of plenoptic [11] and plenacoustic [12, 13] functions, which arise from measuring light and sound, respectively, at various positions in a room. We assume that the position function $h(t)$ has bounded slope, i.e., that for some $\Gamma > 0$, $|dh(t)/dt| \le \Gamma$ pix/s for all t. This corresponds to a bound on the speed at which the object can move in the video, without requiring that this speed be constant. We also assume that the translation signal $h(t)$ is bandlimited, with bandwidth given by Ω_h rad/s.

For any fixed ω_x, we can recognize $e^{-j\omega_x h(t)}$ as a frequency-modulated (FM) sinusoid [14] having carrier

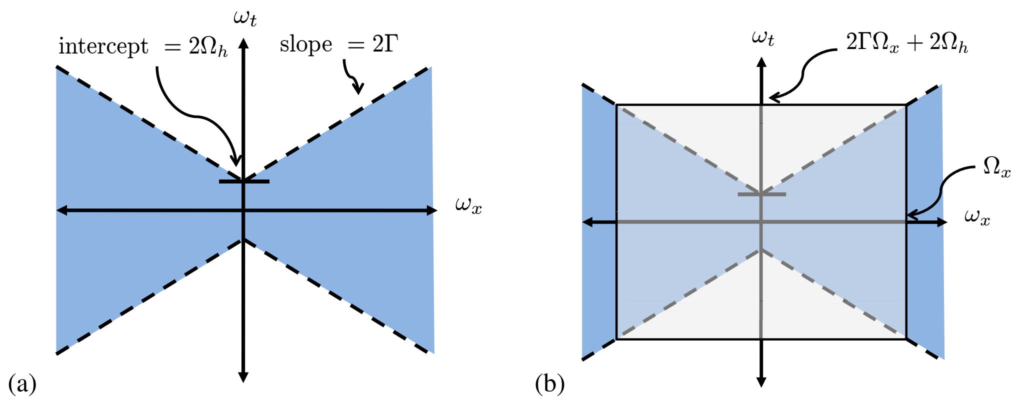


Fig. 1: *(a) Butterfly structure in the two-dimensional Fourier transform of a simple translational 1D video. The slope of the lines is proportional to the maximum speed of the translation. (b) Bandlimiting the video in space (e.g., by spatial lowpass filtering) will also bandlimit the video in time. The resulting temporal bandwidth will be proportional to the spatial bandwidth times the maximum translational speed.*

frequency 0 and instantaneous frequency (in rad/s) $\omega_i(t) = -\omega_x \cdot dh(t)/dt$. Let us also define the deviation term

$$D := \frac{|\omega_x|}{\Omega_h} \max \left| \frac{dh(t)}{dt} \right|.$$

From Carson's bandwidth rule for frequency modulation [14], we have that for fixed ω_x, at least 98% of the total power of $e^{-j\omega_x h(t)}$ must be concentrated in the frequency range $\omega_t \in [-2(D+1)\Omega_h, 2(D+1)\Omega_h]$ rad/s. Since $D \leq \frac{|\omega_x|\Gamma}{\Omega_h}$, we conclude that at least 98% of the total power of $e^{-j\omega_x h(t)}$ must be concentrated in the frequency range $\omega_t \in [-(2|\omega_x|\Gamma + 2\Omega_h), 2|\omega_x|\Gamma + 2\Omega_h]$ rad/s. We note that the dependence of this bandwidth on ω_x is essentially linear.

We conclude that $L(\omega_x, \omega_t)$ will have a characteristic "butterfly shape" with most of its total power concentrated between two diagonal lines that intercept the ω_t-axis at $\pm 2\Omega_h$ and have slope approximately $\pm 2\Gamma$. This shape is illustrated in Figure 1(a). Though not shown, the corresponding figure for the constant velocity model discussed in Section II-B1 would involve a single diagonal line intersecting the origin and having slope $-\Gamma$ (which is half as large as the slope that appears in our more general bounded velocity analysis).

To see the implications of this in terms of bandwidth, let us again suppose that $G(\omega_x)$ is bandlimited (or essentially bandlimited) to the range of frequencies $\omega_x \in [-\Omega_x, \Omega_x]$. We must then have that $F(\omega_x, \omega_t) = G(\omega_x) \cdot L(\omega_x, \omega_t)$ is also (essentially) bandlimited in the spatial direction to the range of frequencies $\omega_x \in [-\Omega_x, \Omega_x]$. Because of the butterfly structure in $L(\omega_x, \omega_t)$, however, this will also cause $F(\omega_x, \omega_t)$ to be (essentially) bandlimited in the temporal direction to the range of frequencies

$$\omega_t \in [-(2\Omega_x\Gamma + 2\Omega_h), 2\Omega_x\Gamma + 2\Omega_h] \text{ rad/s.} \tag{2}$$

This fact, which is illustrated in Figure 1(b), exemplifies a central theme of our work: *filtering a video in the spatial direction can cause it to be bandlimited both in space and in time*. Once again, we note that similar conclusions were reached in the analysis of plenoptic [11] and plenacoustic [12, 13] functions.

3) Sampling implications: From a classical (not compressive) sampling perspective, the Nyquist theorem and the temporal bandwidth predicted in (2) suggest that in order to avoid aliasing, the video should be sampled at a minimum rate of $\frac{2\Omega_x\Gamma+2\Omega_h}{\pi}$ samples/s. Let us plug in some plausible numbers to illustrate the implications of these bounds. First, consider the spatial bandwidth Ω_x of the prototype frame. In a reasonable imaging system, we might expect the pixel size to be balanced with the spatial bandwidth of the frame so that spatial aliasing is avoided. (This should occur naturally if we assume each pixel integrates spatially over a window of size approximately 1 pix.) Thus, one might anticipate that Ω_x will be on the order of π rad/pix. (This corresponds to a spatial bandwidth of $\frac{\pi}{2\pi} = \frac{1}{2}$ cycles/pix, which suggests a spatial Nyquist sample rate of one sample per pix.)

Under the assumption that $\Omega_x = \pi$, (2) suggests the video will have temporal bandwidth limited to approximately $2\pi\Gamma + 2\Omega_h$ rad/s. We note that Ω_h, the temporal bandwidth of $h(t)$, does not depend on the amplitude or slope of $h(t)$, but only on its shape and smoothness. The term Γ, in contrast, increases with the amplitude or slope of $h(t)$, which in turn could increase for objects closer to the camera. We conjecture that in practice (aside from exceptionally non-smooth motions $h(t)$), the $2\pi\Gamma$ term will typically dominate the $2\Omega_h$ term, and therefore *in general a temporal Nyquist sampling rate of roughly* 2Γ *samples/s should suffice to avoid temporal aliasing*. Stated differently, to avoid temporal aliasing we should not allow a moving object to traverse more than $\approx \frac{1}{2}$ pix between adjacent sampling times. While this of course makes strong intuitive sense, we have arrived at this conclusion through a principled analysis—one that illustrates the direct relationship between the speed of object motion in the video and the video's overall temporal bandwidth.

Again, to be clear, the paragraphs above concern the implications of our analysis in classical (not compressive) sampling of a video. As we discuss in Section III, the streaming compressive measurements produced by a single-pixel camera may actually need to be acquired much faster than the video's temporal Nyquist rate (because only one measurement is collected from each frame). The specific reason that the temporal bandwidth of the video will be relevant is because it will impact the spacing of the "anchor frames" that we use to reduce the complexity of the reconstruction problem.

C. Supporting Experiments

1) Within our model assumptions: As an illustration, we can analytically define a continuous-space, continuous-time video that allows us to test our predictions about spectral support. We let the prototype function $g(x) =$

$\text{sinc}\left(\frac{\Omega_x x}{\pi}\right)$, where $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$ and $\Omega_x = \pi$ rad/pix. This definition ensures that $g(x)$ is bandlimited and that its bandwidth equals precisely Ω_x rad/pix. We let the motion signal

$$h(t) = \sum_{i=1}^{5} a_i \text{sinc}\left(\frac{\Omega_h(t-d_i)}{\pi}\right), \tag{3}$$

where Ω_h controls the total bandwidth (we set $\Omega_h = 15$ rad/s), the delays d_i are chosen randomly, and the amplitudes a_i are chosen somewhat arbitrarily but ensure that the maximum value attained by $|h(t)|$ equals some parameter Γ, which we set to 25 pix/s. (By changing Ω_h and Γ, we can independently articulate the bandwidth and the maximum slope of this signal.) Figure 2(a) shows the video $f(x,t)$.

We oversample this video compared to its predicted spatial and temporal bandwidths, and in Figure 2(b) we show the approximate spectrum using the FFT.[1] The blue lines in Figure 2(b) indicate the predicted "butterfly shape" which should bound the nonzero support of $F(\omega_x, \omega_t)$. We see that the empirical spectrum does largely concentrate within this region. For this video, the temporal bandwidth is predicted not to exceed $2\Omega_x\Gamma + 2\Omega_h \approx 187$ rad/s.

In other experiments, we have observed that as we vary the bandwidth and velocity parameters, the approximate support of the estimated spectrum stays within the "butterfly shape" predicted by our theory in Section II-B2. For the sake of space, these experiments are omitted from the current manuscript; however, they are available in a companion technical report [15]. For several videos, we have computed the empirical temporal bandwidth based on our estimated spectrum. To do this, we determine the value of Ω_t for which 99.99% of the energy in the FFT (or windowed FFT) falls within the range $|\omega_t| \le \Omega_t$. In each case, the empirical bandwidth Ω_t equals roughly half of the bandwidth $2\Omega_x\Gamma + 2\Omega_h$ predicted by our theory. (There are occasional exceptions where the unwindowed FFT gives a higher estimate, but this is likely due to sampling artifacts.) This suggests that in some cases, the bandwidth prediction based on Carson's bandwidth rule may be pessimistic.

2) Beyond our model assumptions: Our formal analysis and the experiments described in Section II-C1 pertain specifically to translational videos in which $g(x)$ is bandlimited, $h(t)$ has bounded velocity and bandwidth, and the entire contents of the frame translate *en masse*. However, real world videos may contain objects whose appearance (neglecting translation) changes over time, objects that move in front of a stationary background, multiple moving objects, and so on. We suspect that as a general rule of thumb, the temporal bandwidth of real world videos will be dictated by the same tradeoffs of spatial resolution and object motion that our theory suggests. In particular, the prediction of $2\Omega_x\Gamma + 2\Omega_h$ given by our theory may be approximately correct, if we let Ω_x be the essential spatial bandwidth of the imaging system, Γ be the maximum speed of any object moving in the video, and Ω_h be the essential bandwidth of any object motion. This last parameter is perhaps the most difficult to predict for a given

[1] All spectral plots in Figure 2 show the magnitude of the FFT on a $\log_{10}$ scale. In panels (d) and (f) we apply a smooth Blackman-Harris window to the samples before computing the FFT; this helps remove artifacts from the borders of the sampling region.

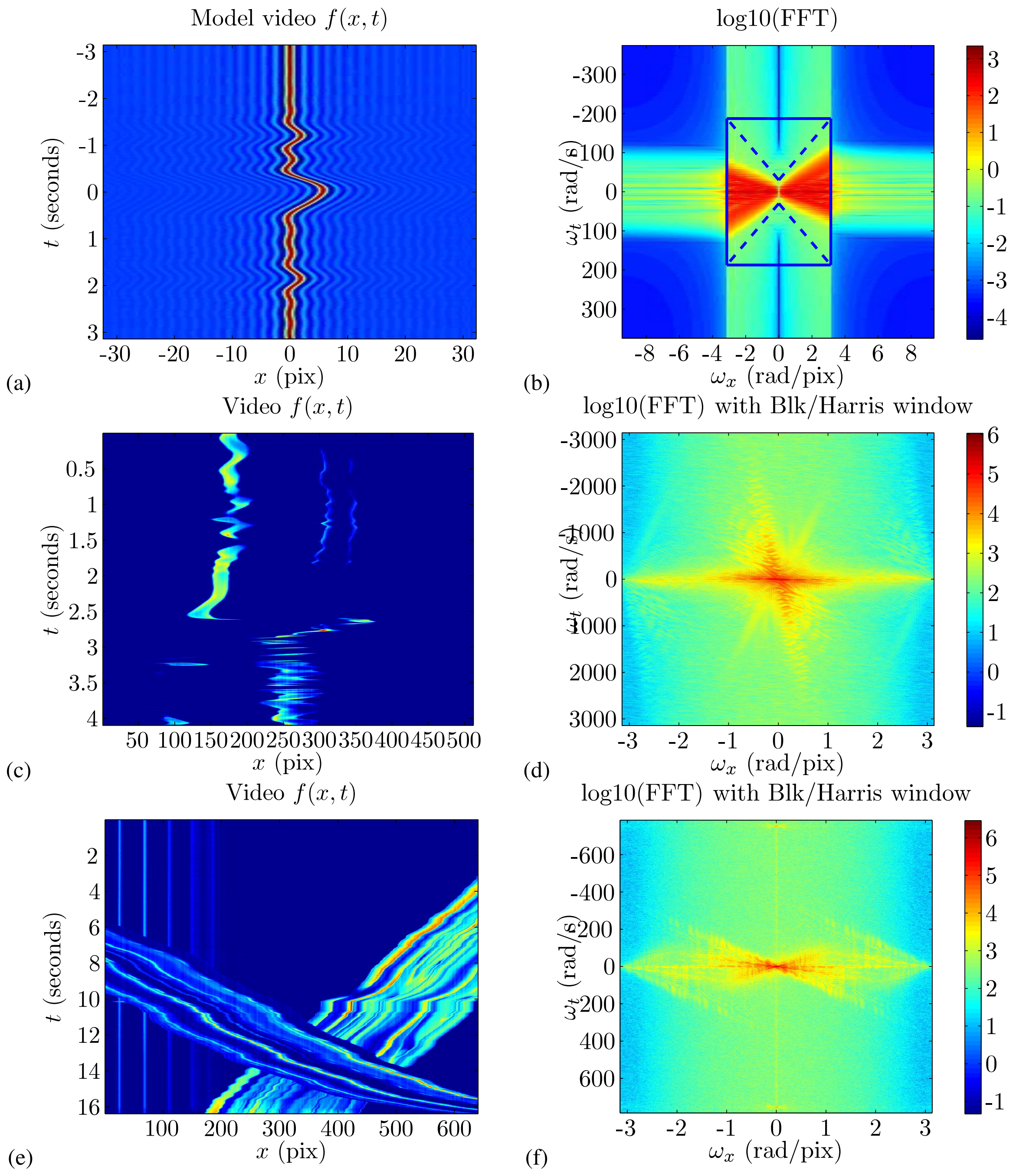


Fig. 2: *Top row: (a) Video $f(x,t) = g(x - h(t))$ with bandlimited profile $g(x)$ and bandlimited translation signal $h(t)$. (b) Estimated spectrum. Middle row: (c) 4096 sampled 1D rows from* Candle *video. (d) Estimated spectrum with windowing to alleviate border artifacts. Bottom row: (e) 4096 sampled 1D rows from* Pendulum + Cars *video. (f) Estimated spectrum with windowing to alleviate border artifacts.*

video, but we suspect that in many cases its value will be small and thus its role minor in determining the overall temporal bandwidth.

To support these conjectures, we have identified the characteristic "butterfly shape" in experiments where $g(x)$ is not bandlimited (e.g., a Gaussian bump convolved with the unit-step function), where $h(t)$ is not bandlimited (e.g., a triangle wave), where there are multiple moving edges, and where there are occluding objects. Again, these experiments are available in a companion technical report [15]. In general, the estimated spectra continue to follow the predicted butterfly shape. However, it remains an open problem to support this with theoretical analysis. We do note that we believe our results are largely consistent with the classical study by Dong and Atick concerning the statistics of real-world videos [16]. Although the videos in that study had relatively low temporal resolution, Dong and Atick did note a certain radial symmetry to the spectrum (with one term depending on ω_t/ω_x) and observe that at low spatial frequencies the power spectrum will have a strong decay as a function of the temporal frequency.

We conclude this examination with our own series of experiments on real-world videos. These videos (courtesy of MERL) were collected in a laboratory setting using a high-speed video camera, but the scenes being imaged contained natural (not particularly high-speed) motions. For each video, we select a 2D "slice" of the 3D video cube, extracting one spatial dimension and one temporal dimension.

We begin with the *Candle* video which features two candle flames in front of a dark background; the video was acquired at a rate of 1000 frames per second. We select 4096 consecutive time samples from the video, and we extract 512 pixels from a certain row in each frame. Figure 2(c) shows the video, and Figure 2(d) shows the estimated spectrum. Again, we recognize the approximate butterfly shape and an approximate limitation to the video bandwidth, both spatially and temporally. More specifically, we see a collection of lines having various slopes, with the lines passing roughly through the origin. However, there is also a bit of "thickness" near the origin due to a possible Ω_h term. The slopes of these lines match what might be expected based on an empirical examination of the video itself. For example, the fastest motion in the video appears to occur at roughly $t = 2.6$ s, where the candle flame translates to the right with a speed of roughly 1500 pix/s. Consequently, we see a portion of the estimated spectrum oriented along a line with slope of approximately 4000 pix/s. Overall, for this video the empirical temporal bandwidth (in this case, the value of Ω_t for which 99% of the energy in the windowed FFT falls within the range $|\omega_t| \leq \Omega_t$) equals 348 rad/s. This suggests that the video's temporal sampling rate (1000 frames/s) may have been higher than necessary.

Next, we consider the *Pendulum + Cars* video, featuring two translating cars and an occlusion as one car passes in front of the other; the video was acquired at a rate of 250 frames per second. We select 4096 consecutive time samples from the video, and we extract 640 pixels from a certain row in each frame. Figure 2(e) shows the video, and Figure 2(f) shows the estimated spectrum. Once again, we recognize the approximate butterfly shape and an approximate limitation to the video bandwidth, both spatially and temporally, and once again, we see a collection of lines with various slopes, but with a bit of "thickness" near the origin due to a possible Ω_h term. The slopes of these lines match what might be expected based on an empirical examination of the video itself. For example, the maximum slope appears to be on the order of 140 pix/s, while the maximum translational speed of the cars appears to be on the order of 70 pix/s. The overall empirical temporal bandwidth is less than 35 rad/s, and consequently, this video's temporal sampling rate (250 frames/s) may also have been higher than necessary.

D. *Videos with Two Spatial Dimensions*

1) Signal model: Our analysis is easily generalized to the more conventional case of videos having two spatial dimensions. We again use the variable $t \in \mathbb{R}$ to index time in seconds, and we use the variables $x, y \in \mathbb{R}$ to index spatial position on the focal plane in pix. We again consider videos belonging to a simple but representative translational model. Let $g(x, y)$ denote a 2D function of space (one can think of this as a continuous-space "still image"), and consider a continuous-space, continuous-time video $f(x, y, t)$ in which each "frame" of the video consists of a shifted version of this prototype frame. More formally, suppose that $f(x, y, t) = g(x - h_x(t), y - h_y(t))$, where $h(t) = (h_x(t), h_y(t))$ is a function that controls how much (in pix) the prototype frame is shifted in the x- and y-directions at each time step.

2) Fourier setup: Let $F(\omega_x, \omega_y, \omega_t)$ denote the 3D Fourier transform of $f(x, y, t)$, and let $G(\omega_x, \omega_y)$ denote the 2D Fourier transform of $g(x, y)$. Using similar analysis to the above, we will have $F(\omega_x, \omega_y, \omega_t) = G(\omega_x, \omega_y) \cdot L(\omega_x, \omega_y, \omega_t)$, where $L(\omega_x, \omega_y, \omega_t) := \mathcal{F}_t\{e^{-j\omega_x h_x(t) - j\omega_y h_y(t)}\}(\omega_x, \omega_y, \omega_t)$. For fixed ω_x, ω_y, $L(\omega_x, \omega_y, \omega_t)$ equals the 1D Fourier transform of $e^{-j\omega_x h_x(t) - j\omega_y h_y(t)}$ with respect to time, evaluated at the frequency ω_t.

3) Temporal bandwidth analysis: We assume that the position functions $h_x(t)$ and $h_y(t)$ have bounded slope, i.e., that for some $\Gamma_x, \Gamma_y > 0$, $|dh_x(t)/dt| \le \Gamma_x$ pix/s and $|dh_y(t)/dt| \le \Gamma_y$ pix/s for all t. This corresponds to a bound on the "speed" at which the object can move in each direction. We also assume that both translation signals $h_x(t)$ and $h_y(t)$ have bandwidths bounded by Ω_h rad/s.

Using arguments that parallel our 1D analysis (see [15]), we conclude that $L(\omega_x, \omega_y, \omega_t)$ will have a characteristic "polytope hourglass shape." Considering the first octant of the 3D frequency space (in which $\omega_x, \omega_y, \omega_t \ge 0$), most of the total power of $L(\omega_x, \omega_y, \omega_t)$ will fall below (in the temporal direction) a plane passing through the points $(0, 0, 2\Omega_h)$, $(1, 0, 2\Gamma_x + 2\Omega_h)$, and $(0, 1, 2\Gamma_y + 2\Omega_h)$. The other seven octants follow symmetrically.

To see the implications of this in terms of bandwidth, suppose that $G(\omega_x, \omega_y)$ is bandlimited (or essentially bandlimited) to the range of frequencies $(\omega_x, \omega_y) \in [-\Omega_x, \Omega_x] \times [-\Omega_y, \Omega_y]$. We must then have that $F(\omega_x, \omega_y, \omega_t) =$

$G(\omega_x, \omega_y) \cdot L(\omega_x, \omega_y, \omega_t)$ is also essentially bandlimited in the spatial direction to the range of frequencies $(\omega_x, \omega_y) \in [-\Omega_x, \Omega_x] \times [-\Omega_y, \Omega_y]$. Because of the hourglass structure in $L(\omega_x, \omega_y, \omega_t)$, however, this will also cause $F(\omega_x, \omega_y, \omega_t)$ to be essentially bandlimited in the temporal direction to the range of frequencies

$$\omega_t \in [-(2\Omega_x\Gamma_x + 2\Omega_y\Gamma_y + 2\Omega_h), 2\Omega_x\Gamma_x + 2\Omega_y\Gamma_y + 2\Omega_h]. \tag{4}$$

Therefore, we see that filtering such a video in the spatial directions can cause it to be essentially bandlimited both in space and in time.

From a classical (not compressive) sampling perspective, if we expect that both Ω_x and Ω_y will be on the order of π rad/pix, and if we assume that the $2\Omega_x\Gamma_x + 2\Omega_y\Gamma_y$ term will typically dominate the $2\Omega_h$ term, then in typical scenarios, to avoid temporal aliasing we should not allow a moving object to traverse more than $\approx \frac{1}{2}$ pix in any direction between adjacent sampling times. Aside from exceptionally non-smooth motions $h(t)$, we do strongly suspect that the influence of the temporal bandwidth Ω_h will be minor in comparison to the influence of the $2\Omega_x\Gamma_x + 2\Omega_y\Gamma_y$ term, and so *in general a temporal Nyquist sampling rate of* $2(\Gamma_x + \Gamma_y)$ *samples/s will likely serve as a reasonable rule of thumb*. Again, this rule of thumb illustrates the direct relationship between the speed of object motion in the video and the video's overall temporal bandwidth. As we will see in Section III, this bandwidth will impact the spacing of the anchor frames that we use to reduce the complexity of the CS reconstruction problem.

III. Anchor Frames for Reducing Reconstruction Complexity

The insight we have developed in Section II suggests that many videos of interest may indeed be exactly or approximately bandlimited in the temporal direction. For problems involving CS of such videos, this implies that there may be a limit to the "complexity" of the information collected by streaming compressive measurement devices. One way of exploiting this limited complexity draws from classical interpolation identities for bandlimited signals. We briefly review these identities in Section III-A before exploring their applications for CS reconstruction in Section III-B.

A. Sampling and Interpolation Principles

Before considering 2D or 3D video signals, let us first review the basic principles involved in non-compressive sampling and interpolation of a bandlimited 1D signal. Suppose that $f(t)$ is a signal with temporal bandwidth bounded by Ω_t rad/s. The Nyquist theorem states that this signal can be reconstructed from a discrete set of samples $\{f(nT_s)\}_{n\in\mathbb{Z}}$, where the sampling interval $T_s \le \frac{\pi}{\Omega_t}$ seconds. In particular, it holds that

$$f(t) = \sum_{n\in\mathbb{Z}} f(nT_s)\mathrm{sinc}\left(\frac{t - nT_s}{T_s}\right). \tag{5}$$

Instead of actually *reconstructing* the continuous-time signal $f(t)$, a more important consequence of (5) for us will be the fact that, for any $t_0 \in \mathbb{R}$, $f(t_0)$ can be *represented* as a linear combination of the discrete samples $\{f(nT_s)\}_{n\in\mathbb{Z}}$.

With varying degrees of approximation, it is possible to replace the sinc interpolation kernel in (5) with other, more localized kernels. We will write

$$f(t) \approx \sum_{n\in\mathbb{Z}} f(nT_s)\gamma\left(\frac{t-nT_s}{T_s}\right), \tag{6}$$

where $\gamma(t)$ is a prototype interpolation kernel. In addition to the sinc kernel, for which $\gamma(t) = \mathrm{sinc}\,(t)$, other possible choices include the zero-order hold (rectangular) kernel, for which $\gamma(t) = 1$ when $|t| \le \frac{1}{2}$ and $\gamma(t) = 0$ otherwise, the first-order hold (triangular, or "linear interpolation") kernel, for which $\gamma(t) = 1 - |t|$ when $|t| \le 1$ and $\gamma(t) = 0$ otherwise, and a variety of cubic interpolation kernels [17].

In general, a smoother choice for $\gamma(t)$ will better approximate the ideal sinc kernel. However, smoother kernels tend to have wider temporal supports, and it can be desirable in some applications (such as the CS recovery problem discussed below) to limit the temporal support of the kernel. One way to improve the performance of the lower-order, more narrow interpolation kernels is to decrease the sampling interval T_s. However, for our CS recovery problem discussed below, this too will increase the complexity of the recovery algorithm by increasing the number of anchor frames.

For a 2D or 3D video with limited temporal bandwidth, the separability of the Fourier transform implies that the interpolation formulas presented above should hold for each spatial location (i.e., for each pixel). Using the videos discussed in Section II-C, we have confirmed this in experiments evaluating the quality of interpolation as a function of the video properties, interpolation kernel, etc. Again, these experiments are available in a companion technical report [15].

B. CS in Streaming Scenarios

1) Measurement process: To set up the CS problem, consider a continuous-space, continuous-time video $f(x, y, t)$. Let $f_d : \{1, 2, \ldots, N\} \times \mathbb{R} \to \mathbb{R}$ denote a sampled discrete-space, continuous-time version of this video, where for $p = 1, 2, \ldots, N$,

$$f_d(p, t) = f(x_p, y_p, t). \tag{7}$$

In the expression above, N specifies the number of pixels in each sampled frame, and (x_p, y_p) represents the spatial location of pixel number p. Typically, $\{(x_p, y_p)\}_{p=1}^{N}$ will form a 2D grid of points in the focal plane. For notational convenience, we rasterize these pixel values and index spatial position in f_d using only the pixel number p.

We consider an imaging system that collects streaming measurements of the video $f(x, y, t)$ according to the following model. We let T denote a measurement interval (in seconds), and we suppose that one linear measurement is collected from f_d every T seconds. (As we will discuss below, typically T will be much smaller than the Nyquist sampling interval suggested by the video's bandwidth.) Letting $y(m)$ denote the measurement collected at time mT, we can write

$$y(m) = \sum_{p=1}^{N} \phi_m(p) f_d(p, mT) = \langle \phi_m, f_d(:, mT) \rangle, \tag{8}$$

where $\phi_m \in \mathbb{R}^N$ is a vector of random numbers (see Section IV-B for details on the measurement vectors we prescribe), and we use "Matlab notation" to refer to a vector $f_d(:, t) \in \mathbb{R}^N$. In total, we suppose that M measurements are collected. Stacking all of the measurements, we have

$$y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(M) \end{bmatrix} = \begin{bmatrix} \langle \phi_1, f_d(:, T) \rangle \\ \langle \phi_2, f_d(:, 2T) \rangle \\ \vdots \\ \langle \phi_M, f_d(:, MT) \rangle \end{bmatrix} = \underbrace{\begin{bmatrix} \phi_1^T & & & \\ & \phi_2^T & & \\ & & \ddots & \\ & & & \phi_M^T \end{bmatrix}}_{\Phi:\ M \times MN} \underbrace{\begin{bmatrix} f_d(:, T) \\ f_d(:, 2T) \\ \vdots \\ f_d(:, MT) \end{bmatrix}}_{x_d:\ MN \times 1}. \tag{9}$$

Stacking the raw frames into a vector $x_d = [f_d(:, T)^T \ f_d(:, 2T)^T \ \cdots \ f_d(:, MT)^T]^T \in \mathbb{R}^{MN}$, and letting Φ denote the $M \times MN$ block diagonal measurement matrix appearing in (9), we can write $y = \Phi x_d$. This appears to be a standard CS problem, with measurements on the left, unknowns on the right, and a measurement matrix that relates the two. Unfortunately, this is a very difficult CS problem to solve directly: first, the recovery problem is very highly underdetermined, with the number of measurements representing only $\frac{1}{N}$ times the number of unknowns; and second, the size of the recovery problem, with MN unknowns, can be immense. We stress that the frames $f_d(:, T), f_d(:, 2T), \ldots, f_d(:, MT)$ are assumed to be very close together in time (perhaps less than a thousandth of a second apart), and so x_d gets very large very quickly.

2) Simplifying the linear equations: Fortunately, if we assume that the video $f(x, y, t)$ has limited temporal bandwidth, we can simplify this recovery process to some degree. Let us assume that $f(x, y, t)$ has temporal bandwidth bounded by Ω_t rad/s. We note that the temporal bandwidth of f_d will also be bounded by Ω_t rad/s.

Let T_a denote a time interval no greater than the Nyquist limit ($\frac{\pi}{\Omega_t}$ seconds) suggested by the video's bandwidth, and assume that $T_a = VT$ for some integer $V \geq 1$. Then for any integer j, we can apply (6) and write

$$\begin{aligned} f_d(:, jT) &\approx \sum_{n \in \mathbb{Z}} f_d(:, nT_a)\gamma\left(\frac{jT - nT_a}{T_a}\right) = \sum_{n \in \mathbb{Z}} f_d(:, nT_a)\gamma\left(\frac{j}{V} - n\right) \\ &= \begin{bmatrix} \cdots & \gamma\left(\frac{j}{V} - 1\right) I_N & \gamma\left(\frac{j}{V} - 2\right) I_N & \gamma\left(\frac{j}{V} - 3\right) I_N & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ f_d(:, T_a) \\ f_d(:, 2T_a) \\ f_d(:, 3T_a) \\ \vdots \end{bmatrix}, \end{aligned}$$

where I_N denotes the $N \times N$ identity matrix. Therefore,

$$\underbrace{\begin{bmatrix} f_d(:, T) \\ f_d(:, 2T) \\ \vdots \\ f_d(:, MT) \end{bmatrix}}_{x_d} \approx \underbrace{\begin{bmatrix} \cdots & \gamma\left(\frac{1}{V} - 1\right) I_N & \gamma\left(\frac{1}{V} - 2\right) I_N & \gamma\left(\frac{1}{V} - 3\right) I_N & \cdots \\ \cdots & \gamma\left(\frac{2}{V} - 1\right) I_N & \gamma\left(\frac{2}{V} - 2\right) I_N & \gamma\left(\frac{2}{V} - 3\right) I_N & \cdots \\ & \vdots & \vdots & \vdots & \\ \cdots & \gamma\left(\frac{M}{V} - 1\right) I_N & \gamma\left(\frac{M}{V} - 2\right) I_N & \gamma\left(\frac{M}{V} - 3\right) I_N & \cdots \end{bmatrix}}_{\Gamma} \underbrace{\begin{bmatrix} \vdots \\ f_d(:, T_a) \\ f_d(:, 2T_a) \\ f_d(:, 3T_a) \\ \vdots \end{bmatrix}}_{x_a}. \qquad (10)$$

Assuming $\gamma(t)$ has temporal support within some reasonable bound, the matrix Γ above will have size $MN \times (\frac{MN}{V} + O(1))$ and so this allows a dimensionality reduction by a factor of V. In other words, using the interpolation matrix Γ, we can relate the raw video frames (acquired T seconds apart) to a reduced set of what we call *anchor frames*, which are spaced T_a seconds apart. In the equation above, the stack of anchor frames is denoted by x_a, and the length of x_a is approximately V times smaller than the length of x_d.

Putting all of this together, we have

$$y = \Phi x_d \approx \Phi \Gamma x_a, \qquad (11)$$

where

$$\Phi\Gamma = \underbrace{\begin{bmatrix} \cdots & \gamma\left(\frac{1}{V} - 1\right) \phi_1^T & \gamma\left(\frac{1}{V} - 2\right) \phi_1^T & \gamma\left(\frac{1}{V} - 3\right) \phi_1^T & \cdots \\ \cdots & \gamma\left(\frac{2}{V} - 1\right) \phi_2^T & \gamma\left(\frac{2}{V} - 2\right) \phi_2^T & \gamma\left(\frac{2}{V} - 3\right) \phi_2^T & \cdots \\ & \vdots & \vdots & \vdots & \\ \cdots & \gamma\left(\frac{M}{V} - 1\right) \phi_M^T & \gamma\left(\frac{M}{V} - 2\right) \phi_M^T & \gamma\left(\frac{M}{V} - 3\right) \phi_M^T & \cdots \end{bmatrix}}_{M \times (\frac{MN}{V} + O(1))}.$$

We have arrived at a CS problem in which the total number of unknowns has been reduced from MN to $\frac{MN}{V} + O(1)$. Indeed, the largest dimension of any matrix or vector in this formulation is now limited to $\frac{MN}{V} + O(1)$ instead of

MN. Moreover, due to decay in γ, the matrix $\Phi\Gamma$ will be banded, with zeros in all positions sufficiently far from the diagonal. This facilitates storage of the matrix and reconstruction of very long video sequences by breaking the recovery problem into blocks. (However, we consider only reconstructing a single sequence of the video in this paper.)

From the formulation above, we see that it is possible to focus the reconstruction process on recovery of a relatively low-rate stream of anchor frames x_a, rather than the high-rate stream of frames x_d measured by the imaging system. Of course, in simplifying the CS problem, we have changed the very unknowns that must be solved for. In many cases, we believe that it will suffice merely to reconstruct and display the anchor frames themselves; however, we note that the raw video frames x_d can be estimated from the reconstructed anchor frames by using the interpolation equation (10).

If the anchor frames are defined at the video's temporal Nyquist rate, and if there are no additional assumptions made about the video, then one should not expect any temporal correlations to remain among the anchor frames. In many real world settings, however, there will be objects moving within the scene, and the smoothness of the object motion can lead to temporal correlations, e.g., that can be captured via motion-compensated transforms. Thus, in order to impose the strongest possible model on the vector of anchor frames, it may be helpful to look for sparsity in a motion-compensated wavelet transform.[2] In Section IV, we propose one method for doing this while confronting the chicken-or-egg problem.

3) Optimizing the spacing between anchor frames: The quality of the interpolation approximation described in (10) relies on the assumption that $T_a \leq \frac{\pi}{\Omega_t}$. However, in practical settings one may not know Ω_t exactly, and this raises the question of how to properly set T_a, the spacing between anchor frames. We note that when T_a is chosen to be too small a multiple of T (that is, when $V = \frac{T_a}{T}$ is small), the dimensionality of the reduced problem (11) may not be small enough to permit reconstruction. Thus, in practice one should generally aim to choose T_a small enough that the interpolation approximation (10) holds to a reasonable degree of accuracy but large enough that the number of unknowns is sufficiently reduced. Our simulations in Section V-B explore these trade-offs and demonstrate that it can indeed be possible to find a setting for T_a that meets both of these criteria.

IV. Multiscale Reconstruction Algorithm

Our analysis in the previous sections has (*i*) revealed that videos that are sampled by imaging devices (including CS imaging devices) may have limited temporal bandwidth, (*ii*) characterized the tradeoffs between the spatial resolution of the camera, the speed of any moving objects, and the temporal bandwidth of the video, and (*iii*) explained

[2]For promising experiments that involve a simplified version of our algorithm (one that uses anchor frames but not with a motion-compensated wavelet transform), we refer the reader to a companion technical report [15].

how a relatively low-rate stream of "anchor frames" can be used to reduce the complexity of the reconstruction problem. In this section, we build on these insights and propose a complete algorithm for reconstructing a video from streaming compressive measurements. In order to construct an effective sparsifying basis for the video, this algorithm involves a motion-compensated wavelet transform, and in order to confront the chicken-or-egg problem, this algorithm is multiscale, employing anchor frames at a sequence of progressively finer scales and alternating between reconstructing an approximation of the video and estimating the motion vectors.

A. *Problem Formulation*

Following the setup in Section III-B, we suppose that one linear measurement is collected from f_d every T seconds. Using (9), we can write the vector of M measurements as $y = \Phi x_d$. Using (10), we can write $x_d \approx \Gamma x_a$, where x_a is formed by stacking the anchor frames. For the sake of simplicity in formulating our algorithm, we specifically take $x_a = [f_d(:, T_a)^T \ f_d(:, 2T_a)^T \ \cdots \ f_d(:, \frac{M}{V} T_a)^T]^T$, i.e., we truncate the stream of anchor frames at the beginning and end so that we consider exactly $N_a := \frac{M}{V}$ anchor frames, and we adapt Γ at the borders to account for this truncation. To model any interpolation errors, we introduce an error term $e \in \mathbb{R}^M$ and model the collected measurements as $y = \Phi\Gamma x_a + e$.

As a sparsifying basis to be used for reconstructing x_a, we employ the lifting-based invertible motion adaptive transform (LIMAT) [9]. Further details regarding LIMAT are given in Section IV-C (and in [9]), but the key idea is this: Given a set of vectors describing the motion in a video, LIMAT yields a motion-compensated wavelet transform (across the temporal and spatial dimensions) intended to provide a sparse representation of that video. For a given set of motion vectors v, we let $\Psi_L(v)$ denote the resulting LIMAT transform. We will apply LIMAT transforms to videos of various sizes; in all cases, $\Psi_L(v)$ is a square matrix, but its dimension will depend on the context.

If one had access to the anchor frames x_a, it would be possible to examine the video and estimate the vectors v describing motions between frames. Alternatively, if one had access to the motion vectors v, one could write $x_a = \Psi_L(v)\alpha_a$ for some sparse coefficient vector α_a and solve a CS reconstruction problem such as

$$\widehat{\alpha}_a = \operatorname*{argmin}_{\alpha_a} \|\alpha_a\|_1 \text{ s.t. } \|y - \Phi\Gamma\Psi_L(v)\alpha_a\|_2 \le \epsilon \tag{12}$$

to recover α_a and subsequently x_a from the compressive measurements [18]. In order to get around this chicken-or-egg problem, we propose a multiscale reconstruction approach.

B. The Multiscale Approach

Rather than reconstructing the full set of high-resolution anchor frames x_a directly, our algorithm aims to reconstruct a sequence of approximations to the anchor frames. These approximations begin at a low temporal and spatial resolution and progress to successively finer resolutions.

To set our notation, we use the variable s to denote scale, with $s = 1$ denoting the coarsest (low-resolution) scale and $s = S$ denoting the finest (high-resolution) scale for some positive integer S. For each $s = 1, 2, \ldots, S$, let $\mathcal{D}_s$ represent a $2^{2(s-S)}N \times N$ linear operator that, when applied to a video frame containing N pixels, performs a spatial low-pass filtering and downsampling operation to produce a low-resolution frame containing $2^{2(s-S)}N$ pixels. (When $s = S$, $\mathcal{D}_s$ is the identity; when $s = S-1$, $\mathcal{D}_s$ averages over 2×2 blocks of pixels; when $s = S-2$, $\mathcal{D}_s$ averages over 4×4 blocks of pixels; etc.) Recalling that

$$x_a = [f_d(:, T_a)^T \ f_d(:, 2T_a)^T \ \cdots \ f_d(:, N_a T_a)^T]^T \in \mathbb{R}^{N_a N},$$

we define for each $s = 1, 2, \ldots, S$, $N_s := 2^{3(s-S)} N_a N$ and

$$x_{a,s} := [\mathcal{D}_s f_d(:, T_a \cdot 2^{S-s})^T \ \mathcal{D}_s f_d(:, 2T_a \cdot 2^{S-s})^T \ \cdots \ \mathcal{D}_s f_d(:, N_a T_a)^T]^T \in \mathbb{R}^{N_s}.$$

The video $x_{a,s}$ contains $2^{s-S}N_a$ frames, each with $2^{2(s-S)}N$ pixels. It represents a lowpass filtered and downsampled version of the original set of anchor frames x_a. We note in particular that $x_{a,S} = x_a$.

Our reconstruction begins at the coarsest scale with an attempt to estimate $x_{a,1}$. Motivated by our temporal bandwidth analysis, we begin reconstruction at a low spatial resolution for the following reasons:

- Our theory predicts that at low spatial resolutions, the temporal bandwidth of the video will be small. This limits the amount by which the video can change between successive frames and justifies the temporal downsampling in the definition of $x_{a,1}$. This downsampling also leaves us with a relatively small number of frames that must be sparsely represented, and so this sequence can be reconstructed from a small number of measurements.
- The spatial downsampling leaves us with larger pixels in each frame, and so an object that traverses, say, P pixels per second in the high-resolution anchor frames will traverse only $2^{1-S}P$ pixels per second at scale $s = 1$. With relatively slow object motion, we can thus obtain a reasonable estimate of $x_{a,1}$ *without employing motion compensation*, or equivalently, with LIMAT motion vectors v set to $\underline{0}$.

After we have reconstructed an estimate of $x_{a,1}$, it is then possible to compute a preliminary estimate of the motion in the video. Although the resulting motion vectors will have limited spatial and temporal accuracy (since $x_{a,1}$ does), they can be used to perform a motion-compensated reconstruction of $x_{a,2}$.[3] From this point, we iterate (as

[3]Typically we choose S small enough so that the frames in $x_{a,1}$ contain enough pixels to obtain a reasonable estimate of the motion vectors.

detailed below), alternating between video reconstruction and motion estimation, and proceeding to finer and finer scales. Between each pair of adjacent scales, we double the spatial resolution and (as suggested by our analysis) double the temporal resolution as well.

The following pseudo-code is an outline of our algorithm.

- **Step 0:** initialize scale $s \leftarrow 1$ and motion vectors $v_1 = \underline{0}$
- **Step 1:** solve

$$\widehat{\alpha}_{a,s} = \underset{\alpha' \in \mathbb{R}^{N_s}}{\operatorname{argmin}} \|\alpha'\|_1 \text{ s.t. } \|y_s - \Phi_s \Gamma_s \Psi_L(v_s)\alpha'\|_2 \leq \epsilon_s, \tag{13}$$

  where, as explained below, y_s contains all measurements in y that are associated with scales $1, 2, \dots, s$, Φ_s and Γ_s relate these measurements to $x_{a,s}$, and ϵ_s accounts for interpolation error
- **Step 2:** form $\widehat{x}_{a,s} = \Psi_L(v_s)\widehat{\alpha}_{a,s}$
- **Step 3:** if $s = S$, terminate algorithm; otherwise go to Step 4
- **Step 4:** given $\widehat{x}_{a,s}$ use motion estimation to compute the set of motion vectors v_{s+1}
- **Step 5:** set $s \leftarrow s + 1$ and go to Step 1

Our reconstruction algorithm places certain constraints on the measurement functions ϕ_m that can be used. In particular, note that for each $s \in \{1, 2, \dots, S\}$ solving the optimization problem (13) requires a set of measurements y_s that are assumed to obey $y_s = \Phi_s \Gamma_s x_{a,s} + e_s$. To facilitate this, we assume that each measurement $y(m)$ is *associated* with some scale $s \in \{1, 2, \dots, S\}$. A measurement $y(m)$ is said to be associated with scale s if $y(m)$ can be written as $y(m) = \langle \widetilde{\phi}_m, \mathcal{D}_s f_d(:, mT) \rangle$ for some $\widetilde{\phi}_m \in \mathbb{R}^{2^{2(s-S)}N}$. Stated differently, this requires that the frame $f_d(:, mT)$ be measured using a function that is piecewise constant over blocks of size $2^{S-s} \times 2^{S-s}$ pixels, i.e., that frame $f_d(:, mT)$ be measured with limited spatial resolution. This is easily accomplished using programmable measurement devices such as the single-pixel camera. In particular, we suggest using noiselets [19] for measurement functions, as they incorporate both randomness and multiple spatial resolutions.

The measurements y_s that we propose to use for solving (13) are merely a subset of the measurements in y. In particular, for each for $s = 1, 2, \dots, S$, y_s contains all measurements in y that are associated with scales $1, 2, \dots, s$. Thus, measurements from coarser scales are re-used for reconstruction at finer scales, and at the finest scale all measurements are used for reconstruction. Other terms appearing in (13) are Γ_s, an interpolation matrix that relates $x_{a,s}$ to a spatially lowpass-filtered stream of the original high-rate frames $[\mathcal{D}_s f_d(:, T)^T \ \mathcal{D}_s f_d(:, 2T)^T \ \cdots \ \mathcal{D}_s f_d(:, MT)^T]^T$, and Φ_s, a measurement matrix populated with the vectors $\widetilde{\phi}_m$. The term ϵ_s appearing in (13) is to account for interpolation error. In our current simulations we assume that we know the amount of interpolation error as an oracle, and we set ϵ_s accordingly.

The specific number of measurements (call this M_s) that we associate with each scale s is a design parameter. We find it useful to allocate a generous portion of the measurement budget to M_1 for two reasons: (*i*) it is relatively cheap to reconstruct an accurate video at low resolution, and (*ii*) an accurate initial reconstruction is important to obtain an accurate initial estimate of the motion vectors. As we progress in the algorithm to finer scales, the number of unknowns that we are solving for increases by a factor of eight with each increment in scale. Therefore, it is also natural to increase the number of measurements M_s that we associate with each scale. Although the size of the video increases by a factor of eight between one scale and the next, however, the sparsity level may not necessarily increase by the same factor. To get a rough estimate of the increase in sparsity level, consider that within a 3D video, the boundary of a moving object will generally trace out a 2D surface. Using simple box-counting arguments, one would generally expect the sparsity level in an isotropic 3D dictionary (such as a 3D wavelet transform) to increase by a factor of four between adjacent scales. Based on this observation, we believe that an increase of M_s by (very roughly) a factor of four for every increment in scale would be a reasonable choice. Our simulations in Section V-B use measurement rates inspired by this $4\times$ rule of thumb and refined with a small amount of trial and error.

Finally, we comment on the allocation strategy of the multiscale measurements across the M frames, i.e., which of the M frames we associate with which scale. One straightforward allocation strategy would be a random one, e.g., randomly choose M_1 out of the M original frames and associate these with scale 1, randomly choose M_2 out of the $M - M_1$ remaining frames and associate these with scale 2, etc. However, we have found that in doing so, it is usually helpful to ensure that all anchor frames at a given scale s are automatically associated with scale s (or a coarser scale). The remaining (non-anchor) frames are associated randomly.

C. Lifting-based Invertible Motion Adaptive Transform (LIMAT)

At each scale $s = 1, 2, \dots, S$, we use LIMAT [9] as a sparsifying transform for the anchor frames $x_{a,s}$. Recall that $x_{a,s}$ contains $2^{s-S}N_a$ frames, each with $2^{2(s-S)}N$ pixels. To simplify notation in this section, set $n = 2^{s-S}N_a$ and for $k = 1, 2, \dots, n$, let x_k denote frame k from the sequence $x_{a,s}$.

The lifting transform partitions the video into even frames $\{x_{2k}\}$ and odd frames $\{x_{2k+1}\}$ and attempts to predict the odd frames from the even ones using a forward motion compensation operator. This operator, which we denote by $\mathcal{F}$, takes as input one even frame and a collection of motion vectors denoted v_f that describe the anticipated motion of objects between that frame and its neighbor. For example, suppose that x_{2k} and x_{2k+1} differ by a 3-pixel shift that is captured precisely in v_f; then as a result $x_{2k+1} = \mathcal{F}(x_{2k}, v_f)$ exactly. Applying this prediction to each pair of frames and keeping only the prediction errors, we obtain a sequence of highpass residual detail frames (see (14) below). The prediction step is followed by an update step that uses an analogous backward motion compensation operator denoted $\mathcal{B}$ and motion vectors v_b. The combined lifting steps

$$\begin{aligned} h_k &= x_{2k+1} - \mathcal{F}(x_{2k}, v_f) & (14) \\ l_k &= x_{2k} + \frac{1}{2}\mathcal{B}(h_k, v_b) & (15) \end{aligned}$$

produce an invertible transform between the original video and the lowpass $\{l_k\}$ and highpass $\{h_k\}$ coefficients. For maximum compression, the lifting steps can be iterated on pairs of the lowpass frames until there remains only one. Ideally, with perfect motion compensation, the $n-1$ highpass frames will consist only of zeros, leaving only one frame of nonzero lowpass coefficients, and making the sequence significantly more compressible. As a final step, it is customary to apply the 2D discrete wavelet transform (DWT) to each lowpass and highpass frame to exploit any remaining spatial correlations.

In our proposed algorithm, we use block matching (BM) [20] to estimate motion between a pair of frames. The BM algorithm divides the reference frame into non-overlapping blocks. For each block in the reference frame the most similar block of equal size in the destination frame is found and the relative location is stored as a motion vector. There are several possible similarity measures; we use the ℓ_1 norm.

V. Discussion and Simulations

In this section, we conclude by describing the differences between our work and several other important ones in the literature. We also present simulations that demonstrate the performance of our algorithm.

A. Related Work

Our algorithm is intended for reconstructing a video from streaming compressive measurements; specifically, we assume that one compressive measurement is collected at each time instant. (The single-pixel camera is a prototypical example of an imaging device that produces such measurements.) We note that a preliminary version of our algorithm [1] was intended for a different measurement model—one in which multiple measurements are collected from each frame but at a much lower frame rate. For example, 1000 measurements might be collected from each of 30 frames per second. While acquiring such measurements may be possible using other compressive imaging architectures [5, 6], our particular interest in this paper is on how to correctly deal with streaming measurements.

In addition to our preliminary algorithm [1], several others have also been proposed in the CS video literature that incorporate motion estimation and compensation. These are discussed below. We note that none of these algorithms were explicitly designed to handle streaming measurements. However, any of these algorithms could be modified to (approximately) operate using streaming measurements by partitioning the measurements into short groups and assuming that all measurements within a group come from the same frame. Such raw aggregation of measurements actually corresponds to using a rectangular interpolation kernel in our formulation (11). As one would expect, and as we will demonstrate in Section V-B, one can achieve significantly better performance by employing smoother interpolation kernels. To the best of our knowledge we are the first to propose and justify the use of other interpolation kernels for aggregating measurements. We do note that one algorithm from the literature [21] does present a careful method for grouping the measurements in order to minimize the interpolation error when using a rectangular kernel.

One related algorithm [22] is based on the observation that nearby frames should have a sparse or compressible residual when one frame is subtracted from the other. The authors employ motion estimation and compensation to produce a sparse or compressible residual frame even in the presence of fast or global motion. More specifically, let x_1 and x_2 denote two frames in a video sequence. Initial estimates of x_1 and x_2 are obtained via independent reconstruction from the measurements $y_1 = \Phi x_1$ and $y_2 = \Phi x_2$, respectively. Motion vectors are then computed between these frame estimates, and subsequently a motion-compensated frame, x_{mc}, is computed from x_1 and the estimated motion vectors. When motion estimation is accurate, the residual $x_{\mathrm{r}} = x_2 - x_{\mathrm{mc}}$ is likely to be highly sparse or compressible. The authors propose to reconstruct the residual frame, x_{r}, from the measurements $y_{\mathrm{r}} = y_2 - y_{\mathrm{mc}} = \Phi x_2 - \Phi x_{\mathrm{mc}}$. The estimate of the residual frame, $\widehat{x}_{\mathrm{r}}$, can be used to improve upon the previous estimate of x_2 via $\widehat{x}_2 = x_{\mathrm{mc}} + \widehat{x}_{\mathrm{r}}$. This procedure is then carried out for the next pair of frames x_2 and x_3. This method requires that the same matrix Φ be used for measuring each frame. In some cases, the use of such measurement matrices may decrease diversity in the measurements. As an extreme example, if a video contains no motion, the measurements from every frame will be identical.

Another related algorithm [23] has been proposed in the field of dynamic medical resonance imaging (MRI). This algorithm relies on the assumption that the locations of the significant coefficients of each video frame can be estimated via a motion-compensated frame. Similarly to the above, a motion-compensated frame, x_{mc}, is obtained from motion vectors computed between a pair of estimates of the frames x_1 and x_2. The initial estimates of the video frames are computed in a frame-by-frame fashion from their respective random measurements. When motion compensation is accurate, the indices of large coefficients of x_{mc} will provide an accurate prediction of the locations of the significant coefficients of x_2, and this knowledge can be used to reconstruct an accurate estimate of x_2. Once a new estimate of x_2 is formed, this procedure is repeated for the next frame x_3, and so on.

Another related algorithm [24] involves dividing a video sequence into several groups of pictures (GOPs), each of which is made up of a key frame followed by several non-key frames. The authors propose to reconstruct each key frame in a frame-by-frame fashion. Given the estimates of the key frames, estimates of the non-key frames are computed via a technique called motion-compensated interpolation. Refined estimates of the non-key frames are obtained in a frame-by-frame fashion, using the initial motion-compensated frame as the starting point of a gradient projection for sparse reconstruction (GPSR) solver. The authors propose a novel stopping criterion for the GPSR solver that helps find a solution that is not too different from the initial estimate. This procedure is then carried out for the next non-key frame, and so on.

Finally, one other related algorithm involves a dual-scale reconstruction of a video [21]. First, a sufficient number of low-resolution measurements are collected to permit a low-resolution preview of the video to be obtained using simple least-squares. Motion information is then extracted from this preview video, and this motion information is used to help reconstruct the high-resolution video sequence. An optimization problem minimizes the sum of the ℓ_1-norm of the expansion coefficients of each individual frame in an appropriate sparsifying transform subject to a data fidelity constraint. Additionally, the minimization problem is subject to a constraint such as

$$\|\widehat{x}_i(x,y) - \widehat{x}_{i+1}(x+v_x, y+v_y)\|_2 \leq \epsilon, \ \forall i,$$

which ensures that the reconstructed video agrees with the motion vectors estimated from the preview video. While this method can be relatively successful in recovering videos from small numbers of measurements, according to the authors, the algorithm can have difficulty in reconstructing high spatial frequency components.

We would like to point out that all of the above methods require, for each frame, a number of measurements proportional to the sparsity level of that frame (after appropriate motion compensation and a spatial sparsifying transform such as a 2D wavelet transform). This is true simply because three of the above methods [22–24] involve reconstructing the video one or two frames at a time. The fourth of the above methods [21] does involve jointly reconstructing the ensemble of video frames. However, this algorithm still requires a total number of measurements proportional to the sum of the sparsity levels of the individual frames because that quantity is what is minimized in the ℓ_1 optimization procedure.

We argue that a temporal sparsifying transformation can help to decrease the sparsity level of a video signal and thus reduce the number of measurements that must be collected of that video. In particular, for videos with slowly moving objects or videos with stationary backgrounds, temporal redundancies may persist for longer than the duration of one or two frames. The above methods, however, are essentially employing a temporal transform with a temporal support of one or two frames. If one can successfully remove temporal redundancies over a longer temporal support, the overall sparsity level of the video will decrease, and this in turn will reduce the number of measurements that must be collected from each frame.

Some methods for CS video reconstruction have been proposed that do employ a 1D temporal sparsifying transform along with a 2D spatial sparsifying transform. In essence, each of these algorithms corresponds to applying a different 3D sparsifying transform Ψ. One algorithm uses the 3D-DWT for Ψ and reconstructs the entire video all at once [2]. Another approach, termed "C_n" and proposed for compressive coded aperture video reconstruction [25], relies on small inter-frame differences together with a spatial 2D-DWT to produce a sparse representation of the underlying video. Unfortunately, neither of these algorithms involves any motion compensation, and so these 3D transforms will not be successful in sparsifying videos containing any significant levels of motion.

We see that the two classes of methods described above have two distinct strengths. Algorithms in the first class employ motion information in the reconstruction to better reconstruct the video sequence, while algorithms in the second class employ a temporal transformation to remove the temporal redundancies over a longer temporal support. In this paper (and in [1]), we use LIMAT which essentially combines these two strengths: LIMAT performs a full motion-compensated temporal decomposition that, when seeded with accurate motion vectors, can effectively remove temporal redundancies even in the presence of complex motion.

We would like to point out, however, that LIMAT is just one of many possible 3D sparsifying bases and that our algorithm can be modified to use other choices of basis as well. We also note that the frame-by-frame reconstruction methods [22–24] have the advantage of being more computationally efficient as they involve reconstruction problems of a smaller size. This may be an important factor in applications where computational resources are scarce and full temporal decompositions are not practical. We also reiterate that to some degree, the multiscale algorithm we present in this paper is a proof-of-concept inspired by our temporal bandwidth analysis. We see our work as an addition to—not a replacement for—the nascent CS video literature, and we believe that the ideas we expound (such as using anchor frames to reduce the complexity of reconstruction) could be combined with some of the other existing ideas mentioned above.

Finally, to place our work in the proper context, we reiterate the differences between a standard ("non-compressive") video capture system and the CS-based ("compressive") video acquisition strategy discussed in this paper. As we discussed in Section I, the primary advantages of a compressive video system are twofold: first, it reduces the physical burden of measuring the incoming video, as it does not require a complete set of samples to be obtained for every frame, and second, it reduces the computational complexity of the encoding process, as it does not require any spatiotemporal transform to be implemented at the encoder. These advantages do not make a compressive video system appropriate for all situations, however. For example, it is well known in CS that for a given sparse signal, the requisite number of measurements is slightly higher (by a small multiplicative factor) than one would require if the relevant sparse components could be directly extracted from the signal. Standard video capture systems have the advantage of getting to "look at" the fully sampled video before identifying the critical features to be encoded. One would naturally expect, then, that for a given quality level, a traditional video encoder would require fewer bits than a compressive video encoder (one in which the CS measurements were quantized and converted to bits). Standard video capture systems also have a second main advantage: the decoding algorithm can be much simpler than in a compressive video system. Our comments on these points are not unique to our particular CS framework; indeed they are fully consistent with observations made in another recent paper dealing with CS for video [26].[4] For our algorithm, however, a detailed comparison against a traditional encoder in terms of bit rate, power consumption, memory requirements, resiliency to errors, etc., is beyond the scope of this paper.

B. Simulations

In this section we present simulation results for our proposed algorithm. We compare our algorithm (using a linear interpolation kernel) to the 3D-DWT [2] and C_n [25] temporal sparsifying transforms using a rectangular interpolation kernel. We also compare to a modified version of our algorithm that uses LIMAT but with zero-motion vectors. The results demonstrate the benefits of combining motion compensation with a temporal sparsifying transform and the benefits of using a non-rectangular interpolation kernel.

We present results for two different test videos (courtesy of MERL), the *Candle* video and the *Pendulum + Cars* video. These are the same videos that we described in Section II-C2, but we have selected different time samples and cropped the frames to different image regions. Four of the original anchor frames of each video are presented in Figures 3(a) and 4(a). Each video consists of 512 frames; the frames in the *Candle* video are of size $N = 128 \times 128$, while the frames in the *Pendulum + Cars* video are of size $N = 256 \times 256$. The first test video contains two candle flames moving in front of a black (zero-valued) background. This video was acquired with a high-speed camera at 1000 frames per second. The flames warp into different shapes and sway from left to right as the video progresses. The typical speed of the moving edge of a flame is approximately 0.3 pixels per frame. The second video contains a white car that is moving in front of a "Newton's cradle" (a pendulum with several silver metal balls) and a stationary background. As the video progresses, the white car moves from right to left, and the two metal balls at each end of the Newton's cradle swing throughout the video sequence. This video was acquired with a high-speed camera at 250 frames per second. The motion of the car is mostly translational (it moves from right to left at a speed of roughly 0.025 pixels per frame), while the two metal balls swing at a much faster speed of roughly 1–1.5 pixels per frame.

[4]The encoder proposed in [26] also has some philosophical similarities and differences with our work and others we have described. Like our work, the system in [26] can operate with frame-by-frame CS measurements and is intended to exploit spatial and temporal correlations in the video. Unlike our work, however, the system in [26] utilizes a special post-measurement encoding of the CS measurements to exploit temporal correlations. It also relies on raw aggregation (rectangular interpolation) of the measurements.

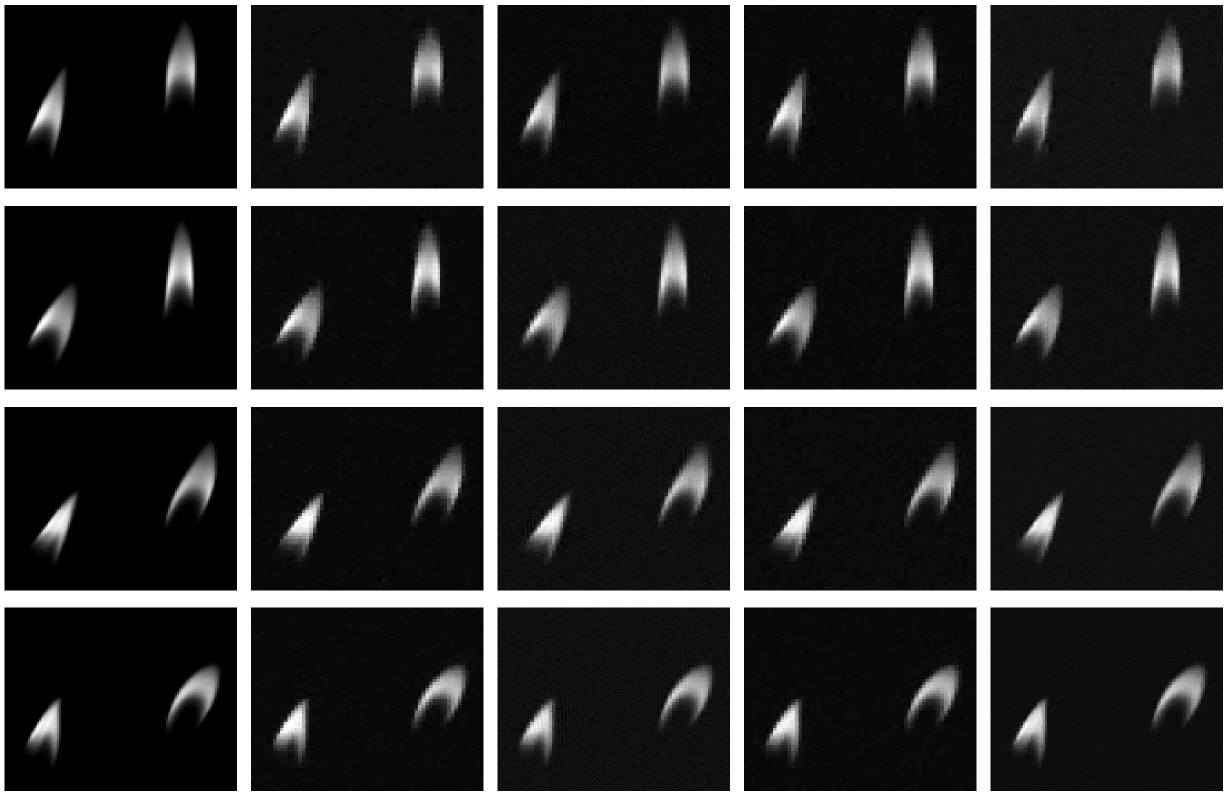
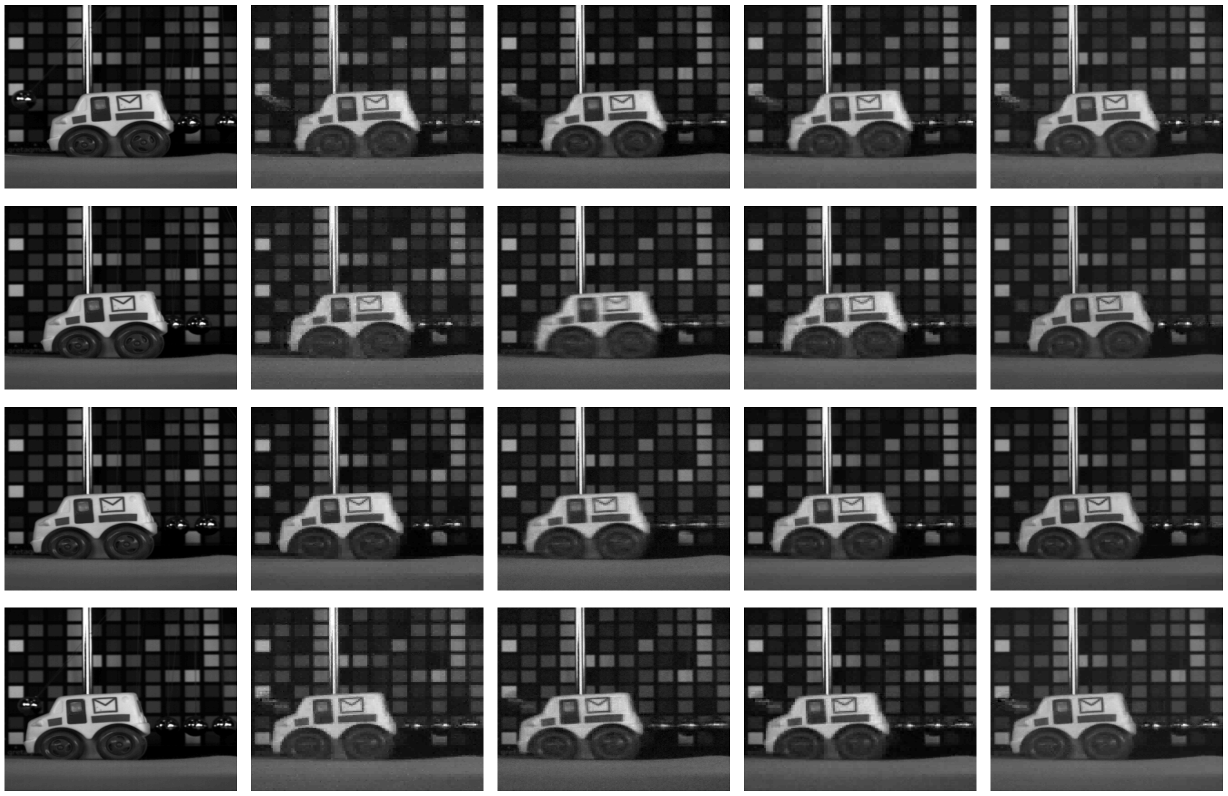


Fig. 3: *Reconstructed anchor frames of* Candle *video. Out of 512 original frames, 32 anchor frames were reconstructed. From the top each row shows the 5th, 11th, 21st, and 28th anchor frame and its reconstruction via various reconstruction methods. From the left each column represents: (a) original frames, (b) 3D-DWT with rectangular interpolation kernel, PSNR* $= 33.67$ *dB, (c)* C_n, $n = 32$, *with rectangular interpolation kernel, PSNR* $= 33.68$ *dB, (d) LIMAT with rectangular interpolation kernel and zero-motion vectors, PSNR* $= 33.80$ *dB, (e) LIMAT with linear interpolation kernel, PSNR* $= 36.76$ *dB.*

In our simulations, we construct synthetic CS measurements from the original high-rate video sequences. In order to reasonably apply CS to either of these video sequences, one would need more than 1000 measurements per second, and thus with the available videos (acquired at 1000 and 250 frames per second) we would not have a sufficient number of measurements if we synthesized only one measurement from each frame. Thus, for the purposes of testing our algorithm, we collect more than one random measurement from each frame of the test video. (Still, the number of measurements we collect from each frame is moderate—120 per frame for the first video and 800 per frame for the second video.) Our setting remains very different from related ones [22–24], however, because we do not intend to reconstruct all of the high-rate frames directly (indeed, this would be difficult to do with so few measurements per frame); rather we will reconstruct a reduced set of anchor frames.

We begin by reconstructing both videos using $N_a = 32$ anchor frames. The total number of unknowns that we are solving for in case of the *Candle* video is $NN_a = 128 \times 128 \times 32$; in the case of the *Pendulum + Cars* video it



(a) Original frames (b) 3D-DWT (c) C_n with $n = 32$ (d) LIMAT with $v = 0$ (e) LIMAT

Fig. 4: *Reconstructed anchor frames of* Pendulum + Cars *video. Out of 512 original frames, 32 anchor frames were reconstructed. From the top each row shows the 4th, 13th, 22nd, and 30th anchor frame and its reconstruction via various reconstruction methods. From the left each column represents: (a) original frames, (b) 3D-DWT with rectangular interpolation kernel, PSNR* $= 28.22$ *dB, (c)* C_n*,* $n = 32$*, with rectangular interpolation kernel, PSNR* $= 29.58$ *dB, (d) LIMAT with rectangular interpolation kernel and zero-motion vectors, PSNR* $= 29.81$ *dB, (e) LIMAT with linear interpolation kernel, PSNR* $= 31.38$ *dB.*

is $NN_a = 256 \times 256 \times 32$. Note that the use of anchor frames reduces the reconstruction complexity by a factor of $V = 512/32 = 16$. For the *Candle* video, we divide the reconstruction into $S = 3$ scales. At scale 1 we reconstruct a low-resolution video of size $N_1 := 32 \times 32 \times 8$, at scale 2 a low-resolution video of size $N_2 := 64 \times 64 \times 16$, and at scale 3 the full-resolution video of size $N_3 := 128 \times 128 \times 32$. Similarly, we divide the reconstruction into $S = 4$ scales for the *Pendulum + Cars* video. Thus, at scale 1 we reconstruct a low-resolution video of size $N_1 := 32 \times 32 \times 4$, at scale 2 a low-resolution video of size $N_2 := 64 \times 64 \times 8$, at scale 3 a low-resolution video of size $N_3 := 128 \times 128 \times 16$, and at scale 4 the full-resolution video of size $N_4 := 256 \times 256 \times 32$. For the *Candle* video we take 120 measurements from each of the 512 frames, and for the *Pendulum + Cars* video we take 800 measurements from each of the 512 frames. Out of the 512 frames in the video, for the *Candle* video, we allocate 54 frames to scale 1, 250 frames to scale 2, and 208 frames to scale 3. Thus, we have $M_1 = 54 \times 120$, $M_2 = 250 \times 120$, and $M_3 = 208 \times 120$ when reconstructing the video. Similarly, for the *Pendulum + Cars* video we allocate 5 frames to scale 1, 30 frames to scale 2, 130 frames to scale 3, and 347 frames to scale 4, so that $M_1 = 5 \times 800$, $M_2 = 30 \times 800$, $M_3 = 130 \times 800$, and $M_4 = 347 \times 800$ when reconstructing the video.

This means that when reconstructing the *Candle* video, at scale 1 we solve for N_1 unknowns using M_1 measurements, and we have $M_1/N_1 \approx 0.79$. At scale 2, we solve for N_2 unknowns using $M_1 + M_2$ measurements, and we have $(M_1+M_2)/N_2 \approx 0.56$ (recall that we re-use coarser scale measurements at the current scale). Finally, at scale 3, we solve for N_3 unknowns using $M_1+M_2+M_3$ measurements, and we have $(M_1+M_2+M_3)/N_3 \approx 0.12$. Similarly, for the *Pendulum + Cars* video, at scale 1 we have $M_1/N_1 \approx 0.98$,[5] at scale 2 we have $(M_1 + M_2)/N_2 \approx 0.85$, at scale 3 we have $(M_1 + M_2 + M_3)/N_3 \approx 0.50$, and at scale 4 we have $(M_1 + M_2 + M_3 + M_4)/N_4 \approx 0.20$.

Figures 3 and 4 show the original anchor frames and their reconstructions via the various reconstruction methods. The PSNR values shown in both figures have been computed across the 32 anchor frames. For each reconstruction method that we test, we use the same multiscale measurements described above; the only differences are in what kernel is used for interpolating the measurements and in what algorithm (i.e., sparsifying transform) is used for reconstruction. Columns (b) and (c) show the performance of the 3D-DWT [2] and C_n [25] temporal sparsifying transforms, respectively. For both we use a rectangular interpolation kernel. As discussed in Section V-A, using a rectangular interpolation kernel is equivalent to aggregating the CS measurements and assigning them to the nearest anchor frame. Column (d) shows the performance of a modified version of our algorithm that uses LIMAT with zero-motion vectors and a rectangular interpolation kernel. Column (e) shows the performance of our full algorithm, using a linear interpolation kernel and using LIMAT with motion vectors estimated during the multiscale reconstruction procedure.

Using the same total number of measurements, we repeat the experiments above but using $N_a = 16$ anchor frames instead of 32 (we also use a different allocation of measurements across the scales). The results are presented in Figures 5 and 6. For all reconstruction methods—including ours—we see a decrease in the PSNR of the reconstruction. We believe that this is mainly due to the inevitable increase in interpolation error that occurs when the anchor frames are spaced further apart. In exchange for the decrease in PSNR, however, the complexity of the reconstruction problem has been reduced by a factor of 2.

In all experiments, we see that our algorithm gives the highest PSNR value. Our reconstruction algorithm also successfully recovers some features in the video frames that other methods cannot. For example, we can see that the flames in column (e) of the *Candle* video reconstructions have sharper and more detailed edges than those of other reconstructions shown in columns (b)–(d). We can also see that the edges of the car and the details of features

[5]Here we could consider taking a few more measurements so that $M_1/N_1 \geq 1$ and simply inverting the measurement matrix to obtain a low-resolution estimate of the video. However, as the measurements contain error due to interpolation, we believe it would be preferable to solve an ℓ_1-regularized problem instead.

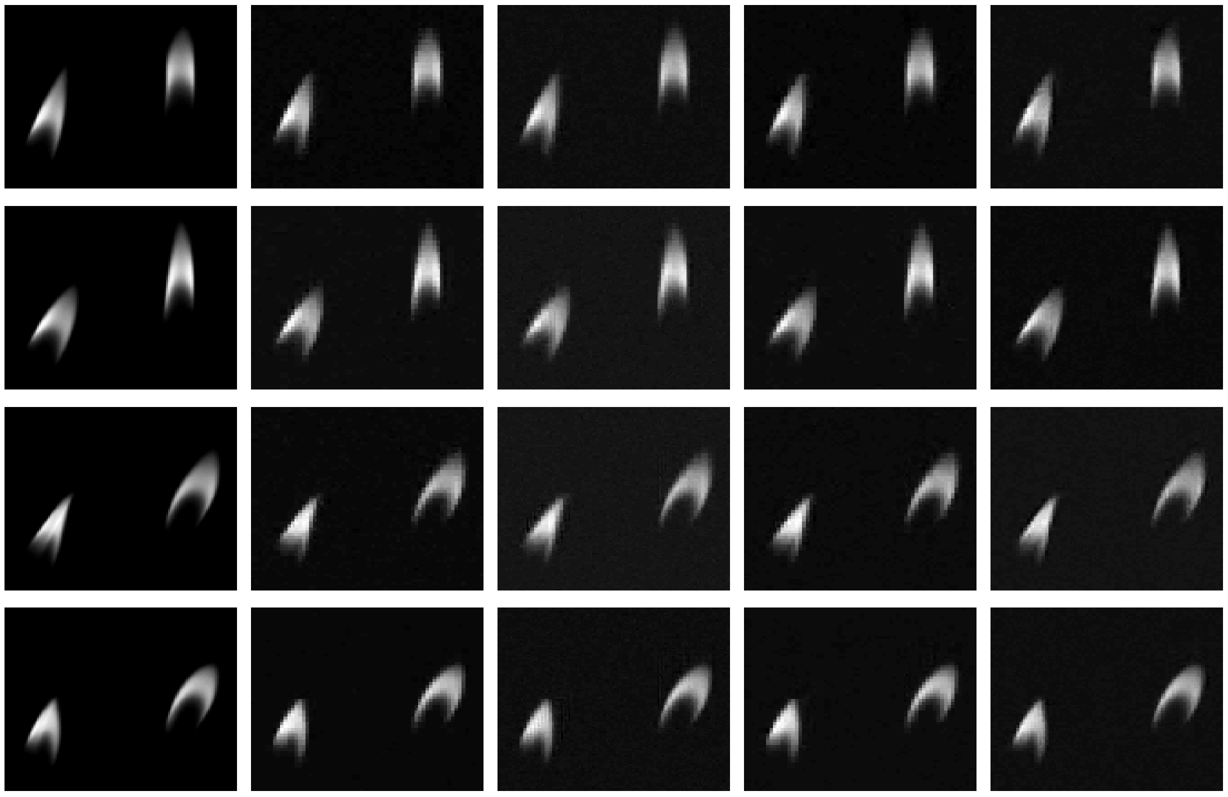


Fig. 5: *Reconstructed anchor frames of* Candle *video. Out of 512 original frames, 16 anchor frames were reconstructed. From the top each row shows the 3rd, 6th, 12th, and 15th anchor frame and its reconstruction via various reconstruction methods. From the left each column represents: (a) original frames, (b) 3D-DWT with rectangular interpolation kernel, PSNR* $= 32.79$ *dB, (c)* C_n, $n = 32$, *with rectangular interpolation kernel, PSNR* $= 32.70$ *dB, (d) LIMAT with rectangular interpolation kernel and zero-motion vectors, PSNR* $= 32.64$ *dB, (e) LIMAT with linear interpolation kernel, PSNR* $= 35.49$ *dB.*

of the wheels are more pronounced than those of the others. We note that objects with slow motion (e.g., cars) are most accurately reconstructed, while objects that are moving fast (e.g., metal balls) are less accurately reconstructed. A different spacing of anchor frames could help to reconstruct more rapidly moving objects, and one could even consider adaptively changing the spacing of the anchor frames throughout the video. While such extensions are beyond the scope of this paper, our companion technical report [15] does contain additional discussion on this front.

Overall, both visually and in terms of PSNR, we have seen the benefits of combining motion compensation with a temporal sparsifying transform and the benefits of using a non-rectangular interpolation kernel. Though not shown, we have also tested our algorithm using zero-motion LIMAT with a linear interpolation kernel. For $N_a = 32$, the PSNR for the reconstructed *Candle* video is 36.60 dB, and the PSNR for the reconstructed *Pendulum + Cars* video is 30.79 dB; for $N_a = 16$, the PSNR for the reconstructed *Candle* video is 34.84 dB, and the PSNR for the reconstructed *Pendulum + Cars* video is 29.35 dB. Thus, while both the motion compensation and the interpolation

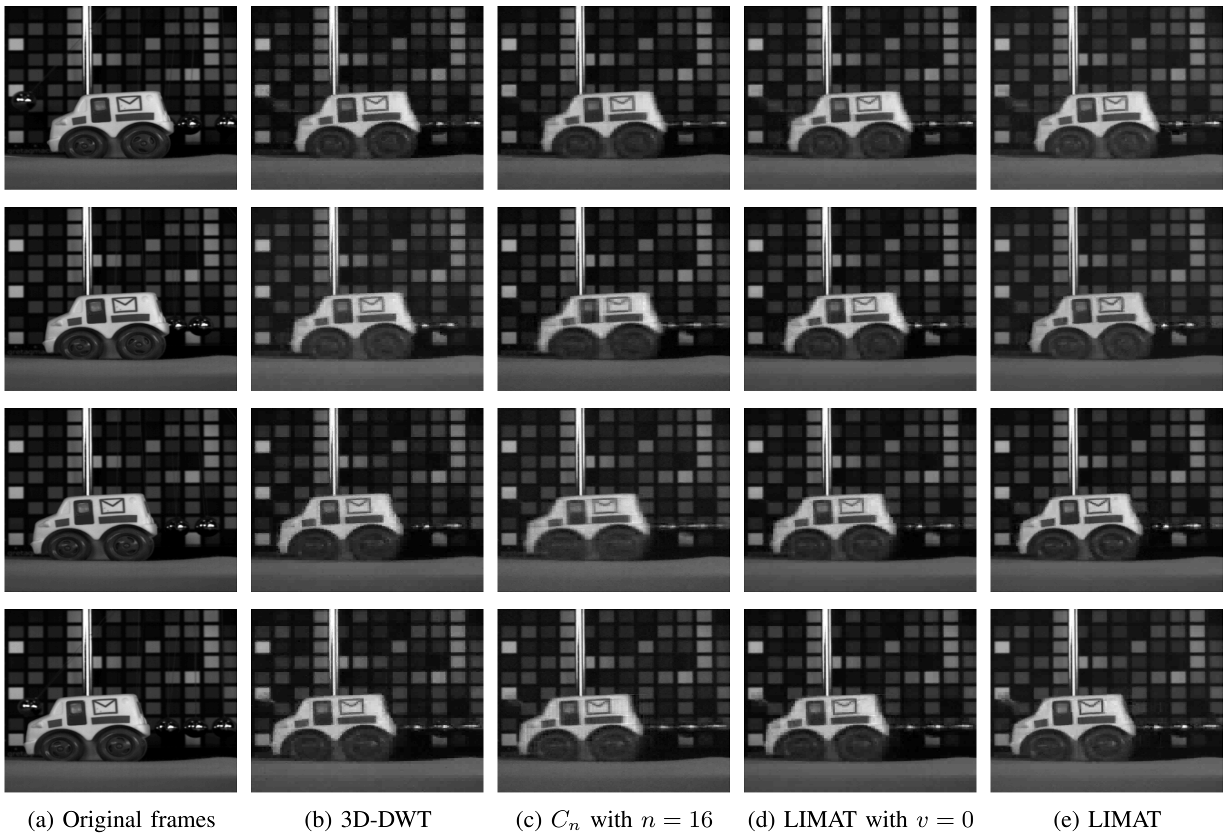


Fig. 6: *Reconstructed anchor frames of* Pendulum + Cars *video. Out of 512 original frames, 16 anchor frames were reconstructed. From the top each row shows the 2nd, 6th, 11th, and 15th anchor frame and its reconstruction via various reconstruction methods. From the left each column represents: (a) original frames, (b) 3D-DWT with rectangular interpolation kernel, PSNR* $= 27.81$ *dB, (c)* C_n, $n = 32$, *with rectangular interpolation kernel, PSNR* $= 28.30$ *dB, (d) LIMAT with rectangular interpolation kernel and zero-motion vectors, PSNR* $= 28.48$ *dB, (e) LIMAT with linear interpolation kernel, PSNR* $= 29.64$ *dB.*

kernel are contributing to the gains that we see, the change in interpolation kernel seems to be a bit more valuable.

The reconstruction time of our algorithm depends mainly on size of the unknown signal. Thus, the reconstruction will be faster at coarser scales in the algorithm and slower at finer scales. The reconstruction complexity also depends on the motion vectors, since the time to compute a matrix vector product increases with the number of non-zero motion vectors used in LIMAT. Thus, the reconstruction time when using LIMAT with zero-motion vectors will be comparable to the time required for, e.g., the 3D-DWT (which is based on orthonormal linear filters) but shorter than the time required for LIMAT with non-zero motion vectors. Using an off-the-shelf Windows-based PC with quad-core CPU, the total times for our algorithm to reconstruct 16 and 32 anchor frames of the *Candle* video were roughly 30 minutes and 230 minutes, respectively. To be more specific, for the reconstruction involving 32 anchor frames, the algorithm required 9 seconds at scale 1, 552 seconds at scale 2, and 13679 seconds at scale 3. For the sake of comparison, the 3D-DWT, C_n, and LIMAT with zero-motion vector algorithms took approximately 2204

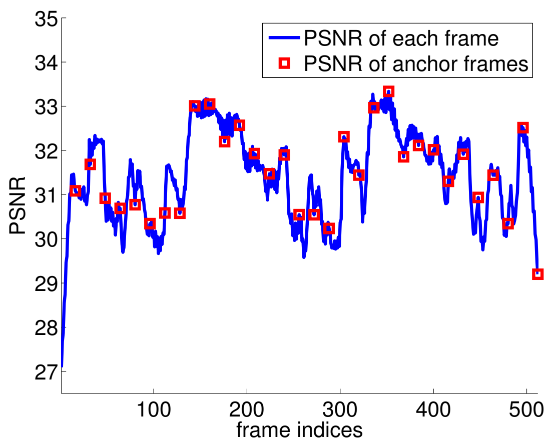


Fig. 7: *Frame-by-frame PSNR values of high-rate* Pendulum + Cars *video estimate. This video was obtained by interpolating the 32 anchor frames that were reconstructed with our algorithm. Four of the reconstructed anchor frames are illustrated in Fig. 4(e). The blue line shows the PSNR values of each frame, and the red squares mark the PSNR values of each anchor frame. The PSNR of the full reconstructed video sequence is* 31.31 *dB, which is only slightly lower than the PSNR of the reconstructed anchor frames.*

seconds, 16106 seconds, and 2796 seconds, respectively. Reducing the computational complexity of our algorithm is an important problem for future research. We do note, though, that while our algorithm may not currently be feasible for real-time reconstruction purposes, as with CS-MUVI [21] it would be possible to view a low-resolution preview of the video in almost real-time.

Finally, although we have reported results above in terms of the reconstruction quality of the anchor frames themselves, it is worth reiterating that from the estimated anchor frames $\widehat{x}_a$ one can, if desired, also obtain an estimate of the original high-rate video sequence via $\widehat{x}_d = \Gamma \widehat{x}_a$. As an example, we construct an estimate of the full 512-frame *Pendulum + Cars* video sequence by interpolating the 32 anchor frames that were reconstructed in Figure 4. In Figure 7, we plot in blue the PSNR for each of the 512 reconstructed frames. Among these 512 frames, of course, are the 32 anchor frames that were reconstructed in the first place; the PSNR for each of these anchor frames is marked with a red square. We note that the PSNR of the full reconstructed video sequence is 31.31 dB, which is only slightly lower than the PSNR of the reconstructed anchor frames. This confirms that the anchor frames capture much of the critical information in the video, and of course reconstructing the anchor frames was an essential step in recovering the full video, because the full video (with $256 \times 256 \times 512$ unknowns) would be very difficult to reconstruct directly from just 800×512 measurements.

Acknowledgments

Thanks to Aswin Sankaranarayanan and Richard Baraniuk for helpful discussions during the collaborative ONR project that partially supported this research. Thanks also to Amit Ashok for providing a helpful reference, and thanks to MERL for providing the real-world videos used in these experiments.

REFERENCES

[1] J. Y. Park and M. B. Wakin, “A multiscale framework for compressive sensing of video,” in *Proc. Picture Coding Symposium (PCS)*, 2009.

[2] M. Wakin, J. Laska, M. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. Kelly, and R. Baraniuk, “Compressive imaging for video representation and coding,” in *Proc. Picture Coding Symposium (PCS)*, 2006.

[3] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, “Single-pixel imaging via compressive sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[4] S. Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 2008.

[5] R. Robucci, J. D. Gray, L. K. Chiu, J. Romberg, and P. Hasler, “Compressive sensing on a CMOS separable-transform image sensor,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1089–1101, 2010.

[6] R. F. Marcia, Z. T. Harmany, and R. M. Willett, “Compressive coded aperture imaging,” in *Proc. SPIE Electronic Imaging*, 2009.

[7] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” *IEEE Trans. Circ. Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, July 2003.

[8] J. Jain and A. Jain, “Displacement measurement and its application in interframe image coding,” *IEEE Trans. Commun.*, vol. 29, no. 12, pp. 1799–1808, Dec 1981.

[9] A. Secker and D. Taubman, “Lifting-based invertible motion adaptive transform framework for highly scalable video compression,” *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1530–1542, Dec. 2003.

[10] Y. Wang, J. Ostermann, and Y. Q. Zhang, *Video processing and communications*. Prentice Hall, 2002.

[11] M. Do, D. Marchand-Maillet, and M. Vetterli, “On the bandwidth of the plenoptic function,” *IEEE Transactions on Image Processing*, vol. 21, no. 2, Feb. 2012.

[12] T. Ajdler, L. Sbaiz, and M. Vetterli, “The plenacoustic function and its sampling,” *IEEE Transactions on Signal Processing*, vol. 54, no. 10, pp. 3790–3804, 2006.

[13] ——, “Dynamic measurement of room impulse responses using a moving microphone,” *The Journal of the Acoustical Society of America*, vol. 122, no. 3, pp. 1636–1645, 2007.

[14] R. E. Ziemer and W. H. Tranter, *Principles of Communication: Systems, Modulation and Noise*. Wiley, 2009.

[15] M. B. Wakin, “A study of the temporal bandwidth of video and its implications in compressive sensing,” Colorado School of Mines Technical Report 2012-08-15, available at http://inside.mines.edu/~mwakin/papers/blvideo-mbw-tr-2012-08-15.pdf.

[16] D. W. Dong and J. J. Atick, “Statistics of natural time-varying images,” *Network: Computation in Neural Systems*, vol. 6, no. 3, pp. 345–358, 1995.

[17] E. Meijering and M. Unser, “A note on cubic convolution interpolation,” *IEEE Transactions on Image Processing*, vol. 12, no. 4, pp. 477–479, April 2003.

[18] E. Candès, “The restricted isometry property and its implications for compressed sensing,” *Compte Rendus de l’Academie des Sciences, Paris*, vol. 346, pp. 589–592, 2008.

[19] R. Coifman, F. Geshwind, and Y. Meyer, “Noiselets,” *Appl. Comput. Harmon. Anal.*, vol. 10, no. 1, pp. 27–44, 2001.

[20] A. Barjatya, “Block matching algorithms for motion estimation,” 2004, technical report, Utah State University.

[21] A. C. Sankaranarayanan, C. Studer, and R. G. Baraniuk, “CS-MUVI: Video compressive sensing for spatial-multiplexing cameras,” in *IEEE Intl. Conference on Computational Photography*, Seattle, WA, April 2012.

[22] J. E. Fowler, S. Mun, and E. W. Tramel, “Block-based compressed sensing of images and video,” *Found. Trends Signal Process.*, vol. 4, no. 4, pp. 297–416, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1561/2000000033

[23] R. Sundaresan, Y. Kim, M. S. Nadar, and A. Bilgin, “Motion-compensated compressed sensing for dynamic imaging,” *Proc. SPIE*, vol. 7798, 2010.

[24] L.-W. Kang and C.-S. Lu, “Distributed compressive video sensing,” in *Proc. Int. Conf. Acoustics, Speech, Signal Proc. (ICASSP)*, 2009.

[25] R. Marcia and R. Willett, “Compressive coded aperture video reconstruction,” in *Proc. European Signal Processing Conf. (EUSIPCO)*, 2008.

[26] S. Pudlewski, A. Prasanna, and T. Melodia, “Compressed-sensing-enabled video streaming for wireless multimedia sensor networks.” *IEEE Trans. Mob. Comput.*, vol. 11, no. 6, pp. 1060–1072, 2012.