REAL-TIME ADAPTIVE VIDEO COMPRESSION

HAYDEN SCHAEFFER *, YI YANG †, HONGKAI ZHAO ‡, AND STANLEY OSHER §

Abstract. Compressive sensing has been widely applied to problems in signal and imaging processing. In this work, we present an algorithm for predicting optimal real-time compression rates for video. The video data we consider is spatially compressed during the acquisition process, unlike many of the standard methods. Rather than temporally compressing the frames at a fixed rate, our algorithm adaptively predicts the compression rate given the behavior of a few previous compressed frames. The algorithm uses polynomial fitting and simple filters, making it computationally feasible, and easy to implement in hardware. Based on numerical simulations of real videos, the algorithm is able to capture object motion and approximate dynamics within the compressed frames. The adaptive video compression improves the quality of the reconstructed video (as compared to an equivalent fixed rate compression scheme) by several dB of PSNR without increasing the amount of information stored as seen in numerical simulations presented here.

Key words. Compressive Sensing, Video Compression, Adaptive Polynomial Fitting, Extrapolation, Optical Flow, Patch Based Methods

AMS subject classifications. 94A08, 65Y99, 94A12

1. Introduction.

Adaptive temporal compression is at the frontier of applications of compressive sensing (CS) [5], making it possible to acquire a large range of scenes using dynamic compression rates. Compressive systems focus on obtaining and storing the least amount of information while still maintaining a high level of recovery. For videos this means removing spatial and temporal redundancy, *i.e.* the high variation of physically observed motion, which appears over different time scales commonly found in video data. Simply stated, we wish to accelerate the acquisition process when the video is static and decelerate when the scene contains dynamic components – all in real-time.

In terms of hardware, current CS methods use physical techniques to code pixel data in order to compress spatial or spectral information. This is commonly done by coded apertures (typically consisting of mechanical gratings or variable materials) which block incoming light in a either patterned or random fashion, thereby subsampling the incoming signal. The idea of CS has had many applications to both hardware and data collection, which include but are not limited to the coded aperture snapshot spectral imaging (CASSI) [31, 32], single pixel camera [12, 33], cooperative analog and digital signal processing transform imager (CADSP) [18], random lens imaging [14], compressive structured light [17], compressive phase retrieval [10, 29], photodetector array camera and spectrometer [30], sparse magnetic resonance imaging (MRI) [28], and many more.

Mathematically speaking, the general forward model for CS can be formulated as follows: if the compression is encoded in a CS matrix A (normally non-invertible), then the relation between the encoded or compressed signal, vector X, and the "original" signal, vector F, is given by $X = AF$, where the assumption is that the data is obtained via linear measurements. For video compressive sensing (VCS) [22, 36, 19, 15, 26, 35], this F contains each frame of the true video, the X denotes the spatially

*,
†
‡
§

and temporally compressed data, while the A contains the random frame by frame masks as well as the temporal compression via a linear combination of frames.

The idea of VCS is vastly different, both mathematically and philosophically, from the classical compression methods. In standard video compression algorithms, the incoming signal is sensed (acquired) in full. The acquired data is then processed, through various transformations and operations, until the data is represented in a sparse way (the compressed video). For example in MPEG-IV, the first frame is compressed in the wavelet basis and stored [22]. Then each incoming frame is stored by compressing the difference between the new frame and the first frame in the wavelet basis. Once the difference exceeds a specific tolerance, the process is reset. In some sense, this type of compression is *sensing then compressing*.

The problem we consider in this paper is that the incoming data exceeds the storage capacity, so the data must be compressed during the acquisition process. For this reason, we call this *video compressive sensing*.

Although there are many works in the literature focusing on the spatial compression of data, the field of variable temporal compression rates is fairly new. There are many potential gains in developing systems and procedures incorporating adaptive temporal compression rate. In terms of the memory, variable compression rates lead to optimized storage space without loss of quality as compared to taking a moderate to high fixed rate. In terms of cost, the resource and energy savings outweigh the computational cost of predicting the frame rate, thereby increasing the efficiency of the system. VCS can be readily applied to many of the common big data sets, for example surveillance videos [4, 25] and traffic data.

In this work, we propose a simple and flexible real-time method to predict frame rates based on adaptive patchwise polynomial fitting. Our method is easy to implement and computationally inexpensive since it is based on temporal differences and polynomial fitting as well as robust to different applications and data conditions. The algorithm can be made parallel and can be extended to different imaging modalities. One current application of this method could be to coded aperture compressive temporal image (CACTI) systems [26].

This paper is organized as follows. Section 1.1 details the data acquired via video compressive sensing. A derivation of our patch based motion estimator is provided in Section 2, with theoretical connections to classical methods. Section 3 details the algorithm and also provides some connections between our model and the underlying physical behavior captured in the video. In Section 4, numerical simulations on real data are provided, which demonstrate the improvement in quality of the reconstructed video given our compression scheme. This section also discusses the robustness of our algorithm on the data acquisition process and the manner in which our algorithm adapts to the data. Lastly we conclude with some final remarks in Section 5.

1.1. Description of the Incoming Data. In VCS each compressed frame $X_j \in \mathbf{R}^{N\times M}$ is a coded linear combination of several true frames $F_{i|j} \in \mathbf{R}^{N\times M}$ with $0 < i \leq T_j$ (where $i|j$ is the i^{th} frame in the j^{th} sequence and where T_j is the frame rate for the j^{th} encoded sequence), *i.e.*

$$X_j := \sum_{i=1}^{T_j} A_{i|j} F_{i|j} \tag{1.1}$$

where $A_{i|j} \in \mathbf{R}^{N\times M}$ is a random binary spatial mask and we take the product above to be element-wise. Although other spatial compression operators can be used, we

only consider binary spatial masks. In practice, either each mask $A_{i|j}$ is independently generated or only the first mask $A_{1|1}$ is randomly generated and each subsequential mask is a fixed translation of the previous one [26]. In this way, X_j has both missing data and motion blur.

The main methodology of adaptive temporal compression is to give an estimate to the most restrictive motion present in the most recent coded frames, and to use this velocity to determine the potential frame rate. In an ideal case, the motion of objects in a video, *i.e.* the optical velocity V between frames, can be calculated using the classical methods of optical flow [20, 6, 1]. From the optical velocity, it is clear that an optimal compression rate T can be determined by the relationship $T \sim \frac{1}{V}$. Due to the corruption, direct application of optical flow or block-matching techniques [15, 21, 13] to estimate V is only possible after reconstructing each frame $F_{i|j}$. However, this drastically increases the computational cost, thus limiting its use in real-time video capturing. Parallel work [35] applies block matching directly on the raw data X_j to get a rough estimate of the fast moving blocks.

1.2. Contribution of the work. The main contribution of this work is the construction of an algorithm which only uses patch based information and simple extrapolation tools. It is necessary to use easy to implement tools in order to allow the algorithm to be incorporated in hardware and to be used in real-time. Our main observation is that as objects enter or leave a given patch, the mean value of the patch changes by an amount related to their speed. In fact, we can show that the speed of the means of the patches is directly related to the optical velocity by the following relationship:

$$|\partial_t \mu^P (X(t))| \approx \frac{|V|}{|P|} \|X\|_{TV(P)} \tag{1.2}$$

where $|P|$ is the size of the patch and $||X||_{TV(P)} := \int_P |\nabla X|$ is the total variation (TV) semi-norm of the patch in space (note that this quantity is time dependent). The proof of this relationship is provided in Section 2 and is important to our proposed algorithm.

1.3. Notation. There are several important variables and functions, for a quick reference we provide a list of them here:

- P is a rectangular patch of fixed size $p_1 \times p_2$.
- V is the velocity of the associated patch P.
- T is the temporal compression rate.
- $\mu^P(X(t))$ is the mean of a frame X in the patch P at time t, the spatial dependents of X is suppressed.
- $\mu_L^P(t)$ and $\mu_Q^P(t)$ are the linear and quadratic approximates (respectively) as a function of time associated with patch P.
- $\mu_{op}^P(t)$ is the optimal approximation of the mean in time for P.

2. Mean Patch Dynamics. In this section, we will formally derive the relationship between the mean patch dynamics, $\partial_t \mu$, and the velocity of objects moving between each frame in a sequence, V. Standard optical flow algorithms estimate V directly by comparing pixels or patches within a given window, those algorithms can be thought of as a Lagrangian method, tracing out the motion path. Our model can be considered as an Eulerian based method, since the algorithm fixes the patch location and observes objects flowing through the patch via $\partial_t \mu$.

In the ideal case, we can define $\mu^P(t)$ to be the mean of frame f over patch P at a given time t (not the compressed frame). We assume that at a given frame, future frames can be locally approximated as smoothly generated displacements. Formally we have the following definition.

DEFINITION 2.1. *We say that a set of frames are* ***temporally consistent*** *if they can be generate by a smooth displacement of the initial frame. More precisely, the sequence is generated by two functions* $(f(x,0), D(x,t)) \in C^1 \times C^1([0,T]; BV)$ *where future frames are related by* $f(x,t) = f(D(x,t),0)$.

This formulation is motivated mathematically and physically. The definition above is mainly used as a local approximation to the temporally behavior of the frames. In particular, if we start with a frame f (setting it to $f(x,0)$), the next frame, over some time dt, will be given by $f(x,dt) := f(D(x,dt),0)$, where $D(x,dt)$ is the deformation of the pixels between the two frames over the small time interval. The rest configuration of the deformation is assumed to be the identity. We will also assume $D \in BV$ and $f \in C^1$, which is true for our algorithm since we consider smooth frames with patchwise (possibly discontinuous) motion.

DEFINITION 2.2. *A temporal displacement function* $D(x,t)$ *is* ***velocity dominated****, if its acceleration is smaller than the velocity, in particular* $||\partial_t^2 D(x,t)|| << ||\partial_t D(x,t)||$. *On the other hand if the first and second time derivatives are on the same order then we say* $D(x,t)$ *is* ***accelerated driven*** .

We assume that typically observed motion is well-approximated by these two behaviors. From a mathematical perspective, these conditions reduce the local dynamics and provide sufficient conditions for polynomial approximations. From the physically perspective, the underlying assumption is that the observed motion is regular, which is common for people, cars, natural objects, etc. In the ideal case though, the velocity of the foreground is restricted to locally constant motion, which we make formal in the following definition.

DEFINITION 2.3. *A temporal displacement function* $D(x,t)$ *is* ***piecewise rigid****, if for any* t, *we have* $\nabla \partial_t D(x,t) \equiv 0$ *over each patch. Specifically, we consider* $\partial_t D(x,-)$ *to be in the patchwise constant (a subset of* BV*).*

The definitions above are related to the standard assumptions in optical flow [20, 6, 1] as well as image registration [27, 8, 23, 34, 24]. In fact, we can show that in some limit, our model recovers the first-order optical flow equation:

$$\partial_t f - \nabla f \cdot V = 0$$

for the ideal sequence $f(x,t)$.

THEOREM 2.4. *Let* $f(x,t)$ *be a temporally consistent sequence of frames generated by a velocity dominated displacement. Then the following hold:*

1. *If* $\theta(x)$ *is the angle between* ∇f *and* $\partial_t D(x,t)$ *at* $t = 0$ *in patch* P *and the angle is bounded by* $||\theta||_{L^\infty(P)} < \epsilon$, *then*

$$|\partial_t \mu^P(t)| = \frac{1}{|P|} \int_P |\nabla f(D(x,dt),0)| \ \ |V(x)| \, dx + \mathcal{O}(dt) + \mathcal{O}(\epsilon) \tag{2.1}$$

2. If $D(x,t)$ is also patchwise rigid then

$$|\partial_t \mu^P(t)| = \frac{|V(P)|}{|P|} \|f(x,dt)\|_{TV(P)} + \mathcal{O}(dt) + \mathcal{O}(\epsilon) \tag{2.2}$$

where $V(P)$ is the patch velocity.

3. As $|P| \to 0$, we recover the first-order optical flow equation.

Proof. To show 1. we differentiate the mean of the frame $f(x,t)$ at time dt. First, let $|P|$ be the area of the patch, then by [3, 2] we have

$$\begin{aligned}\partial_t \mu^P(t) &= \frac{1}{|P|}\int_P \partial_t f(x,dt)dx \\ &= \frac{1}{|P|}\int_P \partial_t f(D(x,dt),0)dx \\ &= \frac{1}{|P|}\int_P \nabla f(D(x,dt),0)\cdot \partial_t D(x,dt)dx\end{aligned}$$

evaluated at time dt. Next, we use the assumption that $D(x,t)$ is velocity dominated to expand the time derivative of the displacement $\partial_t D(x,dt) = \partial_t D(x,0) + \mathcal{O}(dt)$. Using this Taylor expansion and the fact that $f \in C^1$ (specifically, the fact that f has bounded derivatives) we have:

$$\partial_t \mu^P(t) = \frac{1}{|P|}\int_P \nabla f(D(x,dt),0)\cdot V(x)dx + \mathcal{O}(dt) \tag{2.3}$$

where we define $V(x) := \partial_t D(x,0)$ for simplicity. Next, from the assumption on the angle between the image gradients and velocity, we have

$$\partial_t \mu^P(t) = \frac{1}{|P|}\int_P |\nabla f(D(x,dt),0)|\ \ |V(x)|\cos(\theta(x))dx + \mathcal{O}(dt) \tag{2.4}$$

Lastly, Equation (2.1) is achieved via the small angle approximation, $\cos(\theta(x)) = 1 - \mathcal{O}(\theta(x)^2)$.

For 2, we can easily see that if $D(x,t)$ is patchwise rigid, then

$$\begin{aligned}\partial_t \mu(f,P) &= \frac{1}{|P|}\int_P |\nabla f(D(x,dt),0)|\ \ |V(x)|\,dx + \mathcal{O}(dt) + \mathcal{O}(\epsilon) \\ &= \frac{|V(P)|}{|P|}\|f(x,dt)\|_{TV(P)} + \mathcal{O}(dt) + \mathcal{O}(\epsilon)\end{aligned}$$

where $V(P)$ is the patch velocity.

And lastly, for 3, to show that the model recovers the optical flow equation recall

$$\frac{1}{|P|}\partial_t \int_P f(x,dt)dx = \frac{1}{|P|}\int_P \nabla f(D(x,dt),0)\cdot V(x)dx + \mathcal{O}(dt) \tag{2.5}$$

At $dt = 0$, we can differentiate under the integral since f is smooth:

$$\frac{1}{|P|}\int_P \partial_t f(x,0)dx = \frac{1}{|P|}\int_P \nabla f(x,0)\cdot V(x)dx \tag{2.6}$$

therefore we have

$$\frac{1}{|P|}\int_P \left(\partial_t f(x,0) - \nabla f(x,0)\cdot V(x)\right)dx = 0 \tag{2.7}$$

By Lebegue differentiation theorem, as $|P| \to 0$ the integrand goes to zero, thus $\partial_t f(x,0) = \nabla f(x,0)\cdot V(x)$ a.e., which is the first-order optical flow equation at $t = 0$. □

REMARK 2.5. *Note that the patchwise rigid restriction in Theorem 2.4 can be relaxed to having $||\nabla V(x,0)||_{L^p}$ small (by Poincaré-Wirtinger inequality).*

Theorem 2.4 provides the mathematical connection between the ideas presented in this work with the classical optical flow and block matching. The assumptions in the theorem are also related to the physical motion of objects in the frame. For example, the assumption on the angle $\theta(x)$ is equivalent to assuming that the velocity field is applied nearly parallel to the gradients of the dynamic objects in the video.

For the quadratic approximation, second order time derivatives of the patch mean must be consider. The following theorem provides the relationship between the $\partial_t^2\mu^P(t)$ and image characteristics.

PROPOSITION 2.6. *Let $f(x,t)$ be temporally consistent sequence of images generated by a acceleration driven displacement then*

$$\partial_t^2\mu^P(t) = \frac{1}{|P|}\int_P V(x)\cdot\nabla^2 f(D(x,dt))V(x) + \nabla f(D(x,dt))\cdot a(x)\ dx + \mathcal{O}(dt)$$

Proof. The proof is very similar to Theorem 2.4, but instead of taking a first order approximation $D(x,dt)$ in time we take a second order approximation: $D(x,dt) = x + \partial_t D(x,0)dt + \partial_t^2 D(x,0)\frac{dt^2}{2} + \mathcal{O}(dt^3)$. Once again, differentiating and expanding yields:

$$\begin{aligned}
\partial_t^2\mu^P(t) &= \partial_t \frac{1}{|P|}\int_P \nabla f(D(x,dt))\cdot\partial_t D(x,dt)dx \\
&= \frac{1}{|P|}\int_P \partial_t D(x,dt)\cdot\nabla^2 f(D(x,dt))\partial_t D(x,dt) \\
&\qquad + \nabla f(D(x,dt))\cdot\partial_{tt} D(x,dt)\ dx \\
&= \frac{1}{|P|}\int_P V(x)\cdot\nabla^2 f(D(x,dt))V(x) \\
&\qquad + \nabla f(D(x,dt))\cdot a(x)\ dx + \mathcal{O}(dt)
\end{aligned}$$

which provides another relationship between the image gradients, physical characteristics, and algorithmic terms. □

Using these approximation, we can see that first and second order temporal approximations relate to different types of physical motion present in video data. The first order approximation gives an estimation of the dynamics:

$$\mu_L^P(t) = \mu_0 + \partial_t \mu \; t \tag{2.8}$$

while the second order approximation yields:

$$\mu_Q^P(t) = \mu_0 + \partial_t \mu \; t + \frac{\partial_t^2 \mu}{2} t^2 \tag{2.9}$$

In the linear case, the expansion estimates objects which move with velocity dominated motion, *i.e.* $|a| \ll |V|$. On the other hand, the second order approximation gives information on both the average tangential acceleration of objects in the patch as well as the twisting, stretching, and bending forces created by the velocity field. The additional knowledge can give a more appropriate approximation when the objects movement is governed by higher order effects. These assumptions are appropriate for surveillance, tracking, traffic, etc.

3. Compression Algorithm. In this Section 3.1, we provide details on our compression model for VCS which relies on Equation (1.2). We provide some further remarks on the proposed algorithm in Section 3.2. And lastly, to validate our compression results, we will also provide an adaptation of a well-known method for video restoration in Section 3.3, although this is not the focus of our work.

3.1. Our Adaptive Polynomial Fitting. The compression algorithm involves several steps which we summarize below.

1. First we divide each smoothed frame into non-overlapping patches of size $p_1 \times p_2$ in order to capture the local movements, where locality is related to the patch size. The non-overlapping nature breaks the computations down to a decoupled system of small subproblems of the patch sequences (in time), also making parallel computing possible.
2. (Optional): Each of the compressed frames patches are processed by applying an averaging filter with small support (for simplicity we maintain the same notation for the smoothed frame). This removes the anomalies caused by missing data, while preserving the general structures.
3. For a given patch sequence, the mean of each element (denoted by $\mu^P(X)$ for each smoothed frame X and patch P) is calculated.
4. Using the sequence of patch mean, we make an estimate of the optical velocity V via Equation (1.2). and determine if we compress at the extremal ratios or preform adaptive polynomial fitting to estimate the intermediate cases.

The essence of the algorithm is to use $\partial_t \mu^P(X(t))$ as a motion estimator for the optical velocity V instead of directly obtaining V using classical method. This is necessary since we are only able to view the running sum of compressed frames rather than each individual frame in the original video, therefore we can not derive V via the standard optical flow methodology.

To estimate the compression rates in the extremal cases, we can directly use Equation (1.2). If the approximated patch velocity V is very large (small), then the lowest (highest) compression rate is chosen. The approximation of V in Equation (1.2) is a robust estimation of the large and small changes in the patch. Aside from theoretical reasons, we can also see from equation (1.2) that the TV term also helps to mitigate the influences of noisy patches.

In the non-extremal cases, using Equation (1.2) requires specific thresholds relating the optical velocity V to the compression rate T, which is usually much more difficult than deciding the extremal thresholds. In practice, relating V directly to a compression rate requires learning on training data [35]. Seeking a more self-contained estimator, we provide an adaptive approximation of $\mu^P(X(t))$ directly. Since a threshold on the maximum allowable tolerance of the changes in the mean is related to the image intensity and the size of the patch, it provides a less sensitive measure than directly thresholding V. For example, while we can set the tolerance of the change to be a fraction of the maximum image intensity (known data), the tolerance on the velocity must be related to the range of object speeds present in the image (approximated or unknown data).

To adaptively approximate $\mu^P(X(t))$, we use a predictor-corrector like algorithm over a small number of previous frames. For the sake of simplicity, we will restrict the length of the patch sequence to be 4, although the following argument and methodology does not dependent on this value. The given data is now the patch means $\left\{\mu^P(X(t_{j-3})), ..., \mu^P(X(t_j))\right\}$ and their associated time points $\{t_{j-3}, ..., t_j\}$, which are the frame numbers in the true video data. The first three data points of the sequence act as the fitting data, where both a least squares linear fit $\mu^P_L(t)$ and quadratic interpolation $\mu^P_Q(t)$ are calculated (*i.e.* the predictor step). Then using the fourth data point the values $\mu^P_L(t_j)$ and $\mu^P_Q(t_j)$ are compared to the known value $\mu^P(X(t_j))$, providing us with an intrinsic way to learn which polynomial fit to use (*i.e.* the corrector step). Once a fit is chosen, that optimal polynomial $\mu^P_{op}(t)$ is used to estimate the maximum compression rate T such that $\left|\mu^P(t_j + T) - \mu^P_{op}(X_j)\right|$ is within a given tolerance. In the experiments presented in this work, we fixed the tolerance to be 0.15×255, although multiples ranging from 0.1 to 0.2 seem to result in visually comparable results.

In application, the compression rate is usually restricted to a set of fixed values, for example, all the even numbers up to 16. Here we define T_{range} as an increasing sequence of length L storing all the compression rate candidates. We then arrive at the following algorithm for calculating the compression rate T at time point t_j.

In terms of the cost of the algorithm, we consider both the parallel and not parallel implementations. In any individual patch, the complexity is dominated by the averaging filter and patch mean which is $\mathcal{O}(p_1 p_2)$. If the algorithm is run in a parallel environment over K arrays, then the $\frac{MN}{p_1 p_2}$ number of patches can be distributed to $\frac{MN}{p_1 p_2 K}$ patches per array with a total complexity of $\mathcal{O}(\frac{MN}{K})$. Thus our algorithm's complexity is linear in the number of pixels.

A short visual description of the algorithm is detailed in Fig. 3.1. An example of a compressed frame using a frame rate of 4 is shown in Fig. 3.1(a) and its corresponding smoothed version is shown in Fig. 3.1(b). The smoothed version is blurry due to the temporal averaging (frame compression) and the spatial averaging filter. The predicted frame rates for each patch is given in Fig. 3.1(c). In Fig. 3.1(d), the region of predicted motion is highlighted, it contains the car and shadow as well as a few patches from its previous location.

3.2. Further Remarks on our Algorithm. Depending on the data acquisition method, the correction step in the adaptive polynomial fitting can vary. In this paper we consider using the compressed frame $X(t_j)$; however, we can also consider using $A(t_j + 1)F(t_j + 1)$, the first frame in the uncompressed sequence with spatial mask. Since this method can be run in real-time, we can acquire this without calculating the next T. Hence the input data for the fitting can also be chosen as

Algorithm 1 Our Adaptive Temporal Compression Method

Input: $X(t_{j-3})$,...,$X(t_j)$, t_{j-3},...,t_j, p_1, p_2, T_{range}, $V_{\min}$, $V_{\max}$, threshold.
Initialization: (Optional:) Process each X with averaging filter.
Divide each frame into non-overlapping $p_1 \times p_2$ patches. Set $k = 1$.
while $k \leq \frac{MN}{p_1 p_2}$ **do**
Compute $\{\mu(X(t_{j-3})), ..., \mu(X(t_j))\}$ in the k^{th} patch sequence.
Determine the current V from Equation (1.2) with $\mu(X(t_{j-1}))$, $\mu(X(t_j))$, p_1, p_2 and the most recent patch in this sequence.
if $V \leq V_{\min}$ **then**
$T_k = T_{range}(L)$. Break.
else if $V \geq V_{\max}$ **then**
$T_k = T_{range}(1)$. Break.
end if
Calculate $\mu_L^P(t)$ and $\mu_Q^P(t)$ with $\mu(X(t_{j-4}))$, ... , $\mu(X(t_{j-1}))$ and t_{j-4}, ... , t_{j-1}.
Decide the fit $\mu_{op}^P(X(t))$ by comparing the values of $\left|\mu_L^P(t_j) - \mu^P(X(t_j))\right|$ and $\left|\mu_Q^P(t_j) - \mu^P(X(t_j))\right|$.
$T_k = T_{range}(1)$. $i = 1$.
while $i < L$ **do**
if $\left|\mu_{op}^P(t_j + T_k) - \mu^P(X(t_j))\right|$ >threshold **then**
Break.
else
$T_k = T_{range}(i + 1)$. Set $i = i + 1$.
end if
end while
$k = k + 1$.
end while
return $T = \min_k T_k$.

$$\{X(t_{j-2}),\ X(t_{j-1}),\ X(t_j), A(t_j + 1)F(t_j + 1)\}$$

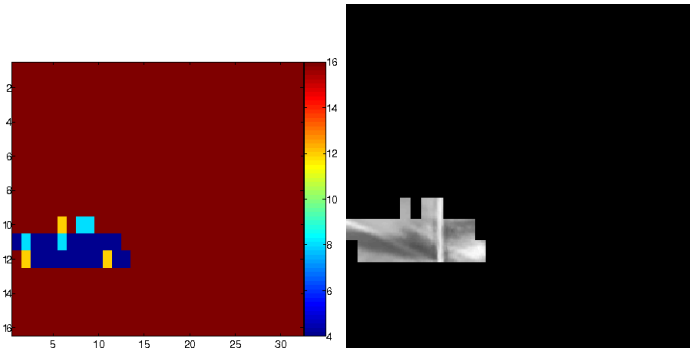
and

$$\{t_{j-2},\ t_{j-1},\ t_j,\ t_j + 1\}.$$

We can also vary the way in which we define the compression rate T. Defining T as the minimum of all the T_k (the predicted compression rate from the k^{th} patch sequence) can be restrictive, allowing outlier values of T_k to dominate in the estimate of T. To avoid this issue, two possible methods can be used. The first is to sort the set $\{T_k\}_k$ and use the ordered data to determine the value T. For example, we could pick the smallest or an average of the pth smallest values. This can be costly, since it may encourage conservative values due to outliers, so instead we introduce another parameter T_{thresh} to relax this minimum. When the ratio of the number of minimum value over $\frac{MN}{p_1 p_2}$ (the cardinality of the T_k set) is smaller than T_{thresh}, we define T as the next compression level in T_{range}, essentially taking the minimum value over a more effective set. By doing so we remove outliers in the data, but we only move up one compression level in order to prevent over estimation. In general, this can be seen as an ordered weighted average (a weighted average on the sorted data set),

(a) Compressed Frame

(b) Average Filter applied to Compressed Frame



(c) Patchwise Predicted Compression Rates

(d) Region of Predicted Motion

FIG. 3.1. *Example of the motion detection element of our algorithm. The compressed frame (a) and the smoothed version (b) depict the input data seen by the algorithm. The region of non-trivial motion detected in the patches is shown in (c) accompanied by the patchwise compression rates in (d). We see that the car and its shadow are the fast moving elements, as expected.*

in which the weights are determined adaptively based on the support of the smallest block-wise compression rate.

3.3. A Restoration Algorithm. In order to verify the success of the forward model, we must also have a way of recovering the compressed frame. Inspired by the reconstruction models from [7, 9], we adapt those previously proposed models to recover the compressed video. Our adapted model for VCS reconstruction is as follows:

$$\min_{F_i} \sum_{i=1}^{T} |DF_i| + \lambda \sum_{i=1}^{T-1} |F_{i+1} - F_i|, \ s.t. \ \sum_{i=1}^{T} A_i F_i = X \tag{3.1}$$

where D is the forward spatial derivatives. Both regularizers in this model are of L^1 type, therefore the minimization can be efficiently solved via the split Bregman method [16].

We first introduce two auxiliary variables G_i for $i = 1, \ldots, T$ and d_i for $i = 1, \ldots, T-1$ and the Bregman variables (constraint enforcing) X^k, B^k and b^k are the

Bregman variables so that the Equation (3.1) becomes:

$$\min_{F,G,d} \sum_{i=1}^{T} |G_i|_1 + \lambda \sum_{i=1}^{T-1} |d_i|_1, \tag{3.2}$$
$$\text{s.t. } G_i = DF_i,\ d_i = F_{i+1} - F_i,\ \sum_{i=1}^{T} A_i F_i = X$$

The constraints are incorporated into the energy as follows.

$$\begin{aligned}
(F^k, G^k, d^k) =& \operatorname*{argmin}_{F,G,d} \sum_{i=1}^{T} |G_i|_1 + \lambda \sum_{i=1}^{T-1} |d_i|_1 \\
&+ \frac{\mu_1}{2} \| \sum_{i=1}^{T} A_i F_i - X + X^{k-1} \|^2 \\
&+ \frac{\mu_2}{2} \sum_{i=1}^{T} \| G_i - DF_i + B_i^{k-1} \|^2 \\
&+ \frac{\mu_3}{2} \sum_{i=1}^{T-1} \| d_i - F_{i+1} + F_i + b_i^{k-1} \|^2 \\
X^k =& \sum_{i=1}^{T} A_i F_i^k - X + X^{k-1} \\
B_i^k =& G_i^k - DF_i^k + B_i^{k-1} \\
b_i^k =& d_i^k - F_{i+1} + F_i + b_i^{k-1}.
\end{aligned}$$

The minimizers for G^k and d^k are explicit:

$$\begin{aligned}
G_i^k =& \text{shrink}(DF_i^{k-1} - B_i^{k-1}, 1/\mu_2) \\
d_i^k =& \text{shrink}(F_{i+1}^{k-1} - F_i^{k-1} - b_i^{k-1}, \lambda/\mu_3)
\end{aligned}$$

where the shrink function is defined for vectors by: $\text{shrink}(\cdot, \tau) := \max(\| \cdot \| - \tau, 0) \frac{\cdot}{\| \cdot \|}$.

For the F variable update, the minimizing equation is the following linear system

$$(\mu_1 A^T A + \mu_2 D^T D + \mu_3 D_3^T D_3) F = \mu_1 A^* (X - X^{k-1}) + \mu_2 D^T (G^k + B^{k-1}) + \mu_3 D_3^T (d^k + b^k) \tag{3.3}$$

where D_3 is the forward difference with respect to the frame (not to be confused with the spatial differences).

Altogether, we alternate the shrinkage steps with a few iterations of conjugate gradient to solve Equation (3.3) in order to find F. The convergence of this algorithm to the correct minimizer is guaranteed, for example see [11].

For the results here, we use the following parameters:

$$\lambda(T) = \begin{cases} 2.67, \text{ if } T = 4 \\ 8.33, \text{ if } T = 8 \\ 25, \text{ if } T = 12 \\ 25, \text{ if } T = 16 \end{cases}$$

$$\mu_1(T) = \begin{cases} 0.67, \text{ if } T = 4 \\ 0.33, \text{ if } T = 8 \\ 1, \text{ if } T = 12 \\ 1, \text{ if } T = 16 \end{cases}$$

$$\mu_2(T) = \begin{cases} 0.167, \text{ if } T = 4 \\ 0.133, \text{ if } T = 8 \\ 0.05, \text{ if } T = 12 \\ 0.1 \text{ if } T = 16 \end{cases}$$

$$\mu_3(T) = \begin{cases} 0.67, \text{ if } T = 4 \\ 1.33, \text{ if } T = 8 \\ 3, \text{ if } T = 12 \\ 5, \text{ if } T = 16 \end{cases}$$

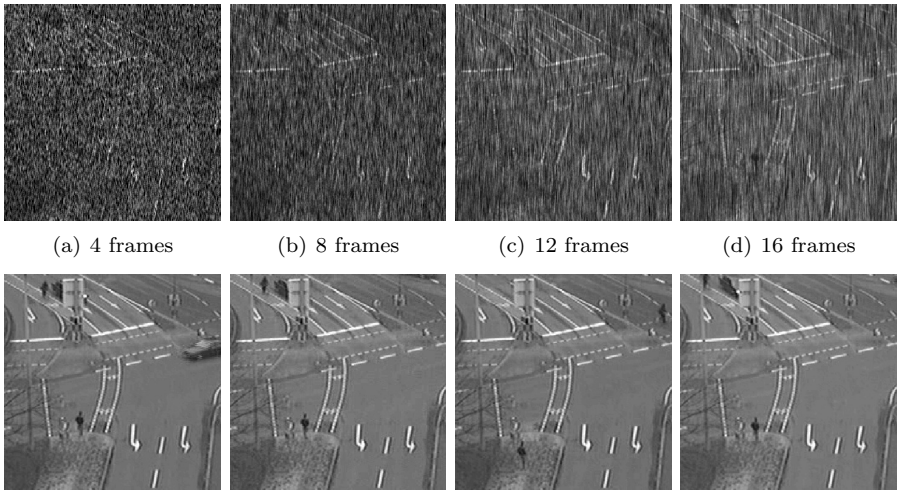
The PSNR of the results are not very sensitive to these parameters, in the sense that a 10% change will not dramatically change the value of the PSNR. To choose the parameters for the reconstruction, we first fit them to a few frames and use the fitted parameters for the entire video sequence.

4. Experimental Results. In this section we use numerical experiments to demonstrate the robustness and effectiveness of our algorithm. As shown in [26], the shifted masks will give comparable reconstruction results compared with completely random masks. Hence in most of our tests we first generate a random binary mask with 50% zeros, and keep shifting it in one direction to get subsequent masks. Other types of masks will also be considered in a later test. In our tests, four different compression rates are considered, 4, 8, 12 and 16.

The experimental results are divided into smaller sections as follows. In Section 4.1, we show that advantage of using variable compression rates depending on the sequence dynamics. In Section 4.2, we demonstrate the gain in using adaptive polynomial fitting rather than only using one type of polynomial. We present some results on the dependence of the algorithm to the patch size in Section 4.3. We also show the gains over using a fixed rate compression algorithm in Section 4.4. And lastly, we apply our algorithm to another type binary mask generation in Section 4.5.

4.1. Video Dynamics and Compression. Fig. 4.1 displays some selected compressed frames using different compression rates. In Fig. 4.1(a), since so few frames are averaged the effective spatial mask appears to be random, while in Fig. 4.1(b-d) as the frame rate increases so does the clarity. However, although the resolution increases with the frame rate, the trade-off is that the image becomes blurred. In essence, this is the balance in adaptive compressive video sensing.

Before we show the performance of our algorithm, we would like to highlight the importance of adaptiveness in video compression through some experiments. Let us first look at how T influences the reconstruction results of different types of video data. Here in each test T frames are compressed into one according to (1.1), then we apply TV-based video reconstruction algorithm on this compressed data to recover



(a) 4 frames (b) 8 frames (c) 12 frames (d) 16 frames

(e) Frame from (a) (f) Frame from (b) (g) Frame from (c) (h) Frame from (d)

FIG. 4.1. *In (a-d), various compressed frames are shown. Each scene is compressed with a different rate and a frame from the scene is displayed in (e-h). The hierarchy shows that the scene with the car suddenly entering in (a) and (e) has the smallest compression rate while the one with pedestrian motion has the highest rate. The medium compression rates, as seen in (b)&(f) or (c)&(g), are related to the car entering in the top left quadrant and the movement of the second pedestrian, respectively.*

TABLE 4.1
Mean PSNR comparison for different types of video data

	Moving frames		Frozen frames	
	T = 4	T=16	T=4	T=16
Test 1	33.7849	28.2370	48.4005	75.3180
Test 2	37.7345	34.9292	45.4085	67.1824

the original frame sequence, and the average PSNR (Peak signal-to-noise ratio) of the reconstructed sequence will be recorded.

Two types of video data are considered in the test, moving frames and frozen frames, where moving frames mean all the T frames are different from each other, while frozen frames stand for the case with T identical frames. The results are recorded in Table 4.1. We can see from the table that when we have stationary video data, a larger T value usually leads to better reconstruction results with higher PSNR values. On the other hand, when there are a lot of movements in the video, a smaller T is often more desirable.

Based on the above observation, we then use numerical tests to check the advantage of adaptive compression over fixed rate compression. In the first setting, we generate a video of 32 frames, where the first 16 frames contain a lot of movements, while the rest are identical. We then manually compress the video into 5 frames, where $T_1 = \ldots = T_4 = 4$ and $T_5 = 16$. According to our assumption on T_{range}, this is the optimal way of adaptive compression for this particular video. We also define another compression by setting $T_1 = 8$ and $T_2 = \ldots = T_5 = 6$, and this is close to the fixed rate compression. For each case, the above reconstruction algorithm (3.1) is applied on these compressed frames to recover each original frame sequence separately,

and the average PSNR of each sequence is recorded.

In the second setting, the same two compression strategies are used on the video with 32 moving frames. The mean PSNR are displayed in the following table. Similar tests are also conducted on videos where the first 16 frames are identical while the rest are moving frames.

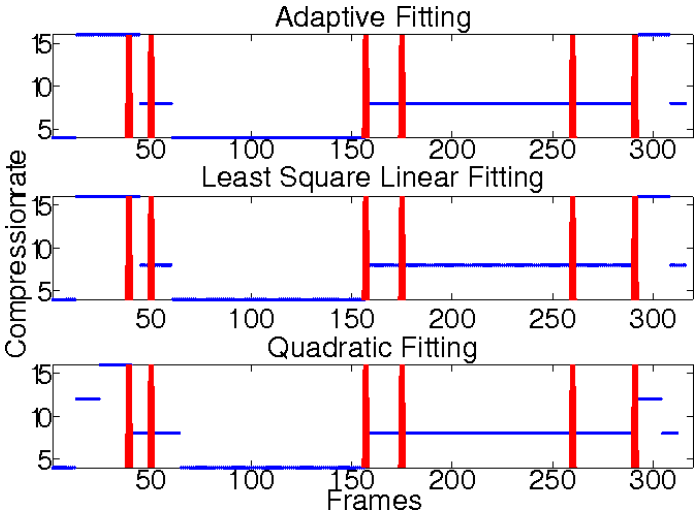
We can see from the PSNR values in Setting 1 that adaptive compression leads to much better reconstruction results. The results in Setting 2 and 3 justify matching the compression with the motion, and not the choice of this particular partition, is responsible for the PSNR gain. In particular, we see that the results of the reconstruction algorithm can support the choice of compression rate as long as there is a significant gain in the PSNR.

4.2. Behavior of Adaptive Polynomial Fitting. In Figure 4.2, the temporal compression rates are plotted for each frame in two real data sets. The red markers indicate changes in the video sequence, for example an object entering or leaving the field of view, while the blue dots stand for the compression rate for each frame. To investigate the effect of our adaptive polynomial fitting step in our algorithm, we compare our method to the case when we restrict the approximating polynomial to be either linear or quadratic only. In Figure 4.2 (a), the algorithm is applied to parking lot surveillance data, containing a static background with moving people and vehicles. The spatial compression rate is taken to be 50%. The first 12 frames are assigned a compression rate of 4 in order to generate input data to our algorithm. The initial computed compression rate is 16 since there is no movement, and decreases to 8 and 4 as the car enters (the first two red markers, where the car starts to enter at the first marker, and fully enters at the second marker). The second two markers are at the frame location when the car gradually stops and people enter the field of view. Since the dynamic component of the video is slowing down, the compression rate should increase, coinciding with our algorithm's performance. At the end of the sequence the moving people become obscure (effectively exiting the field of view) and reappear, which creates a jump in the compression rate. In this case, the adaptive fitting prefers the least squares linear fit, since most of the motion is locally constant.

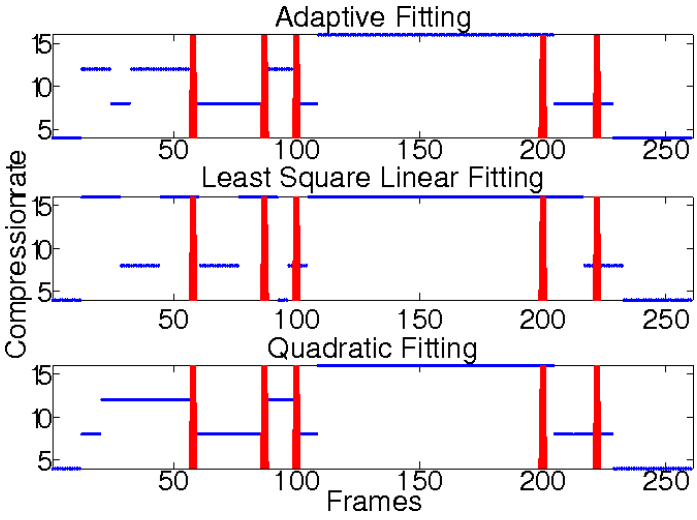
In Figure 4.2 (b), the algorithm is applied to traffic data. Prior to the first marker, the main moving component consists of non-uniform pedestrian movement, therefore a medium-level compression rate is favored. This can be seen visually and agrees with our algorithm's performance. The first two markers shows the occurrence of a vehicle entering the field at various speeds with varying visibility. Therefore, we expect the compression rate to drop. This is exhibited by both the adaptive compression algorithm and the quadratic approximation. The next two markers bound the interval in which the frames are still. And the last signifies a fast moving car entering the video. In this case, the adaptive algorithm incorporates information from the quadratic fitting while also capturing information (near frame 35) that is overlooked using one polynomial exclusively.

TABLE 4.2
Mean PSNR comparison between adaptive and fixed rate compression

	Setting 1		Setting 2		Setting 3	
	Adaptive	Fixed	Adaptive	Fixed	Adaptive	Fixed
Test 1	**53.1740**	40.6072	33.8991	**35.0905**	39.9775	**42.6081**
Test 2	**55.9060**	43.0610	36.5681	**37.2725**	39.7499	**43.2499**



(a) Linear Favored



(b) Combination

FIG. 4.2. *Comparison between adaptive polynomial fitting versus fixing the degree of the polynomial. Using different data (a) and (b) shows that the adaptive polynomial fit may favor one polynomial or use a combination of both.*

In general, the linear fit favors consistent motion, since it approximates one velocity over several compressed frames. The quadratic fit captures more subtle dynamics, shown through its ability to adjust to gradual changes; however, at times this results chooses the more prudent compression rate. Since the approximations are done patch by patch, one interesting observation to note is that the adaptive compression rate does not necessarily give you either the linear or the quadratic fitting result at any given frame. Instead, it balances the contributions from both approximations.

4.3. Comparison of Patch Size. In Figure 4.3, we provide a comparison between different patch sizes. To measure optimality of the patch size, we look at the

qualitative behavior of the compression results and the quantitative results (compression rate and PSNR). The optimal patch size for the parking lot data set is 16 by 8 (with an average PSNR of 45.45) and for this reason it is the patch configuration used in other sections. Figure 4.3 displays the result for patch configurations of 8 by 8 (average PSNR of 41.37), 8 by 16 (average PSNR of 44.96), and 16 by 16 (average PSNR of 45.15).

The patch size of 8 by 16 gives similar results; however, it neglects the motion of the pedestrians. The small square patch of size 8 by 8 is more sensitive to small changes between frames. The larger square patch of size 16 by 16 is less sensitive to small objects. The 16 by 16 patch size results has issues reconstructing the final part of the video data, where the small scale motion is dominant. The 8 by 8 patch size consistently gives too low of a compression rate, since it is sensitive to outliers, thus making it ineffective as a data reduction tool.

From this we can conclude that the patch size is determined by two factors: the scale of motion and its directionality. For consistent velocity V over a sampling time of Δt, one could argue that the diameter of the patch should satisfy the relationship $\text{diam}(P) = V\Delta t$ to capture the correct scale. To correctly resolve directionality, the patch should be shorter in the direction of fast motion and long in the direction of slow motion. For the data set tested here, the patch size of 16 by 8 corresponds to the correct size given this analysis as well. This also shows that since 8 by 16 gives similar results while 8 by 8 gives worse results, scale plays a more important role then directionality.

4.4. Comparison with Fixed Rate Compression. As seen in the tables, reconstruction algorithms for video compressive sensing provide satisfactory results when (and only when) many frames are averaged over low motion scenes or when few frames are averaged during high motion scenes. Therefore, since our algorithm takes advantage of this principle, we would expect that it should yield a better recovered video than using a fixed rate compression. In Fig. 4.4 and 4.5, the compression rate versus frame rate is plotted along with the PSNRs of each frame after applying the reconstruction algorithm. For the results displayed in Fig. 4.4, the mean PSNR is 45.43 dB, compared to a mean of 40.56 dB when applying a fixed rate compression with the same number of compressed frames (*i.e.* identical mean compression rates). The maximum and minimum PSNRs of our algorithm is 77.90 dB and 30.97 dB respectively, while the fixed rate compression yields 55.07 dB and 28.82 dB respectively. For the results in Fig. 4.5, our mean PSNR is 49.58 dB and the fixed rate compression has an average PSNR of 42.92 dB. The maximum and minimum PSNR of our method is 77.20 dB and 30.95 dB, while the fixed rate compression yields 52.67 dB and 28.31 dB, respectively. In both cases, our compression method provides significant gains in the average PSNR of the reconstructed image.

In Fig. 4.6, six reconstructed frames (Frame 93 to 98) from the video used in Fig. 4.4 are displayed. In Fig. 4.7, the same six frames are shown which are reconstructed from data compressed with a fixed rate. The results using our algorithm are of higher quality in the regions containing fast motion than that of a fixed rate. Also, since the temporal compression averages consecutive frames, motion elements in some frames can pollute both the compression and reconstruction of neighboring frames, as seen by the errors incurred around the car in Fig. 4.5. In these figures, it is clear that the adaptive compression method yields less motion and compression artifacts than that of the fixed rate compression, thus resulting in better overall visual quality of the reconstructed video.

4.5. Robustness to Compressive Sensing Matrix. Lastly, we apply our algorithm to the case when the spatial compressive sensing mask is randomly generated for each frame (retaining 45% of the pixels for any given frame). In Fig. 4.8, the mean PSNR is 44.63 dB (39.97 dB for a fixed rate compression) with a maximum and minimum PSNR of 74.98 dB and 30.82 dB respectively (53.77 dB and 28.89 dB for a fixed rate compression). This is comparable with the results from Fig. 4.4, since the algorithm does not depend explicitly on the manner in which the mask is generated. Since the compression algorithm considers average patch information, a particular mask realization should not alter the patch means significantly. This shows the potential of incorporating our algorithm in various video compression applications.

5. Conclusion. We present an adaptive algorithm for predicting compression rates in real-time. The underlying idea is simple: compress more when the video contains little motion and compress less during dynamic scenes. Using this idea, we build an efficient and compact way to estimate the motion of a sequence of compressed and subsampled frames using patch means. By considering the patch means, we reduce the size of the problem and decouple each task. Also, by using each individual patch's history to predict its own compression rate, we accelerate the time it takes to compute the predicted frame rate. Our adaptive model is shown to improve the PSNR of the reconstructed video by several dB as well as the visual quality of the images.

REFERENCES

[1] Gilad Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (4):384–401, 1985.

[2] Luigi Ambrosio and Gianni Dal Maso. A general chain rule for distributional derivatives. *Proceedings of the American Mathematical Society*, 108(3):691–702, 1990.

[3] Luigi Ambrosio and Gianni Dal Maso. On the relaxation in bv $(\omega; \mathbb{R}^m)$ of quasi-convex integrals. *Journal of functional analysis*, 109(1):76–97, 1992.

[4] R Venkatesh Babu and Anamitra Makur. Object-based surveillance video compression using foreground motion compensation. In *Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on*, pages 1–6. IEEE, 2006.

[5] Richard G Baraniuk. Compressive sensing [lecture notes]. *Signal Processing Magazine, IEEE*, 24(4):118–121, 2007.

[6] John L Barron, David J Fleet, and SS Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[7] José M Bioucas-Dias and Mário AT Figueiredo. A new twist: two-step iterative shrinkage/thresholding algorithms for image restoration. *Image Processing, IEEE Transactions on*, 16(12):2992–3004, 2007.

[8] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM computing surveys (CSUR)*, 24(4):325–376, 1992.

[9] H. Jiang C. Li, W. Yin and Y. Zhang. An efficient augmented Lagrangian method with applications to total variation minimization. *Rice CAAM Report TR12-13*, 2012.

[10] Wai Lam Chan, Matthew L Moravec, Richard G Baraniuk, and Daniel M Mittleman. Terahertz imaging with compressed sensing and phase retrieval. *Optics letters*, 33(9):974–976, 2008.

[11] Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. unpublished, 2012.

[12] Marco F Duarte, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin F Kelly, and Richard G Baraniuk. Single-pixel imaging via compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):83–91, 2008.

[13] M Ezhilarasan and P Thambidurai. Simplified block matching algorithm for fast motion estimation in video compression. *Journal of Computer Science*, 4(4):282–289, 2008.

[14] Rob Fergus, Antonio Torralba, and William T Freeman. Random lens imaging. 2006.

[15] Didier J. Le Gall. The mpeg video compression algorithm. *Signal Processing: Image Communication*, 4(2):129 – 140, 1992.

[16] Tom Goldstein and Stanley Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[17] Jinwei Gu, Shree Nayar, Eitan Grinspun, Peter Belhumeur, and Ravi Ramamoorthi. Compressive structured light for recovering inhomogeneous participating media. *Computer Vision–ECCV 2008*, pages 845–858, 2008.

[18] Paul Hasler and David V Anderson. Cooperative analog-digital signal processing. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 4, pages IV–3972. IEEE, 2002.

[19] Yasunobu Hitomi, Jinwei Gu, Mohit Gupta, Tomoo Mitsunaga, and Shree K Nayar. Video from a single coded exposure photograph using a learned over-complete dictionary. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 287–294. IEEE, 2011.

[20] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1):185–203, 1981.

[21] C-H Hsieh and T-P Lin. Vlsi architecture for block-matching motion estimation algorithm. *Circuits and Systems for Video Technology, IEEE Transactions on*, 2(2):169–175, 1992.

[22] Didier Le Gall. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, 1991.

[23] Carole Le Guyader and Luminita A Vese. A combined segmentation and registration framework with a nonlinear elasticity smoother. In *Scale Space and Variational Methods in Computer Vision*, pages 600–611. Springer, 2009.

[24] T Lin, E-F Lee, I Dinov, Carole Le Guyader, P Thompson, AW Toga, and Luminita A Vese. A landmark-based nonlinear elasticity model for mouse atlas registration. In *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, pages 788–791. IEEE, 2008.

[25] Limin Liu, Zhen Li, and Edward J Delp. Efficient and low-complexity surveillance video compression using backward-channel aware wyner-ziv video coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(4):453–465, 2009.

[26] Patrick Llull, Xuejun Liao, Xin Yuan, Jianbo Yang, David Kittle, Lawrence Carin, Guillermo Sapiro, and David J Brady. Coded aperture compressive temporal imaging. *arXiv preprint arXiv:1302.2575*.

[27] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*, 1981.

[28] Michael Lustig, David Donoho, and John M Pauly. Sparse mri: The application of compressed sensing for rapid mr imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

[29] Stefano Marchesini. Ab initio compressive phase retrieval. *arXiv preprint arXiv:0809.2006*, 2008.

[30] Albrecht Poglitsch, Christoffel Waelkens, Norbert Geis, Helmut Feuchtgruber, Bart Vandenbussche, Louis Rodriguez, Oliver Krause, Etienne Renotte, Christiaan Van Hoof, P Saraceno, et al. The photodetector array camera and spectrometer (pacs) on the herschel space observatory. *Astronomy and Astrophysics*, 518, 2010.

[31] Ashwin Wagadarikar, Renu John, Rebecca Willett, and David Brady. Single disperser design for coded aperture snapshot spectral imaging. *Applied optics*, 47(10):B44–B51, 2008.

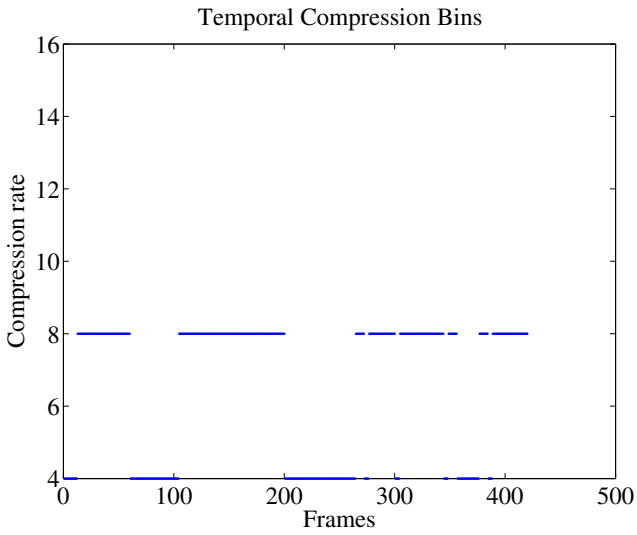
[32] Ashwin A Wagadarikar, Nikos P Pitsianis, Xiaobai Sun, David J Brady, et al. Video rate spectral imaging using a coded aperture snapshot spectral imager. *Opt. Express*, 17(8):6368–6388, 2009.

[33] Michael Wakin, Jason Laska, Marco Duarte, Dror Baron, Shriram Sarvotham, Dharmpal Takhar, Kevin F Kelly, and Richard G Baraniuk. Compressive imaging for video representation and coding. In *Picture Coding Symposium*, 2006.

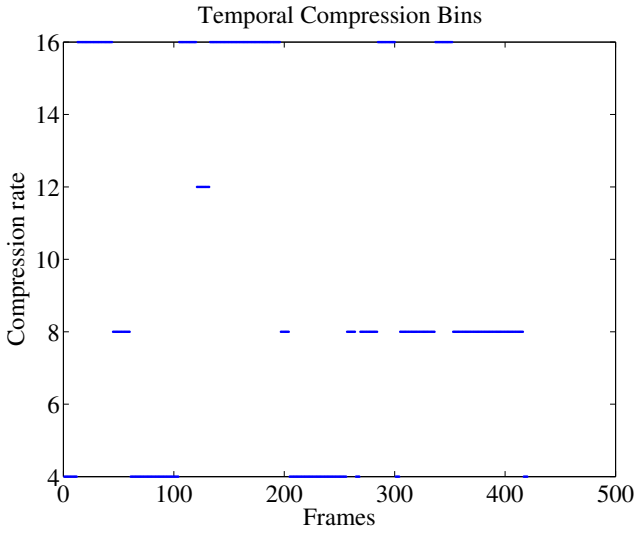
[34] Igor Yanovsky, Carole Le Guyader, Alex Leow, Paul Thompson, and Luminita Vese. Nonlinear elastic registration with unbiased regularization in three dimensions. In *Computational Biomechanics for Medicine III, MICCAI 2008 Workshop*, 2008.

[35] Xin Yuan, Jianbo Yang, Patrick Llull, Xuejun Liao, Guillermo Sapiro, David J Brady, and Lawrence Carin. Adaptive temporal compressive sensing for video. *arXiv preprint arXiv:1302.3446*, 2013.

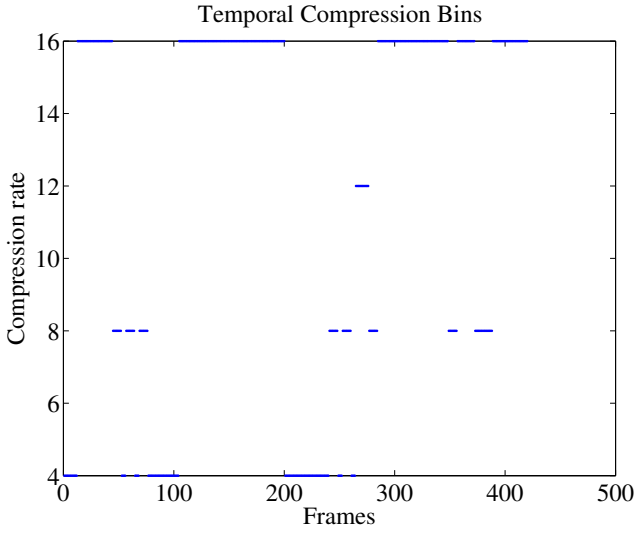
[36] Jing Zheng and Eddie L Jacobs. Video compressive sensing using spatial domain sparsity. *Optical Engineering*, 48(8):087006–087006, 2009.



(a) Patch Size of 8 by 8



(b) Patch Size of 8 by 16



(c) Patch Size of 8 by 16

FIG. 4.3. *Comparison of compression results for various patch sizes using our algorithm.*

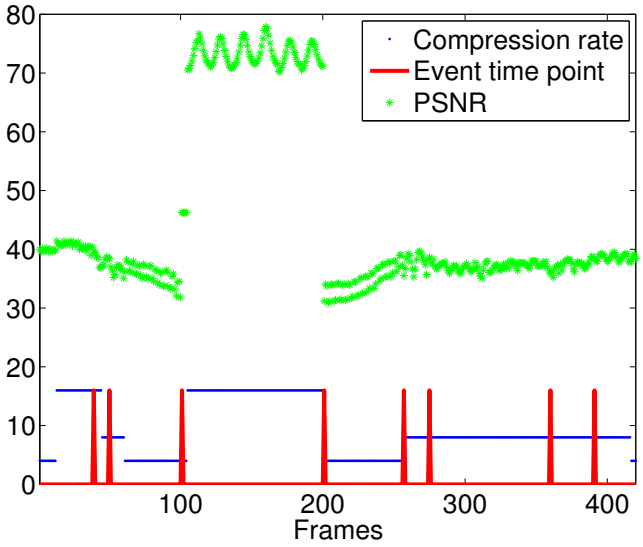


FIG. 4.4. *The compression rates (in blue) and the corresponding PSNRs (in green) of the reconstructed video method to the results generated by our algorithm to the surveillance data set. To increase the range of motion in our video data, we freeze the video at the* 100^{th} *frame and resume the original video at the* 201^{st} *frame.*

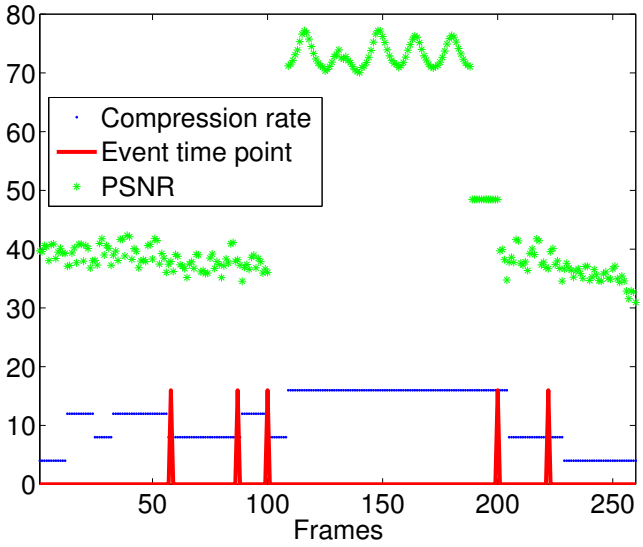


FIG. 4.5. *The compression rates (in blue) and the corresponding PSNRs (in green) of the reconstructed video method to the results generated by our algorithm to the traffic data set.*

FIG. 4.6. *The reconstruction results of six consecutive frames compressed by our algorithm. The PSNRs for the first row starting from the left is: 33.7896 35.7451 36.1793 and the PSNRs for the second row starting from the left is: 33.7118 32.6931 34.7961.*

FIG. 4.7. *The reconstruction results of six consecutive frames compressed using a fixed rate, chosen to match the average compression rate of our algorithm. The PSNRs for the first row starting from the left is: 32.5228 33.9217 34.5397 and the PSNRs for the second row starting from the left is: 33.9049 32.0122 29.9636.*

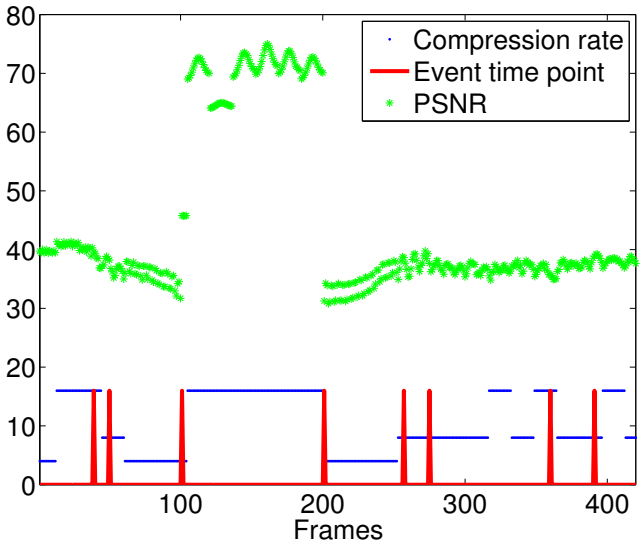


FIG. 4.8. *The compression rates (in blue) and the corresponding PSNRs (in green) of the reconstruction method applied to the results generated by our algorithm to the data that is acquired via a random CS mask. Notice the qualitative and quantitative similarity to Fig. 4.4.*