Real Time Compressive Sensing Video Reconstruction in Hardware

Garrick Orchard, Jie Zhang, Yuanming Suo, Minh Dao, Dzung T. Nguyen, Sang Chin, Christoph Posch, Trac D. Tran, and Ralph Etienne-Cummings, *Fellow, IEEE*

***Abstract*—Compressive sensing has allowed for reconstruction of missing pixels in incomplete images with higher accuracy than was previously possible. Moreover, video data or sequences of images contain even more correlation, leading to a much sparser representation as demonstrated repeatedly in numerous digital video formats and international standards. Compressive sensing has inspired the design of a number of imagers which take advantage of the need to only subsample a scene, which reduces power consumption by requiring acquisition and transmission of fewer samples. In this paper, we show how missing pixels in a video sequence can be estimated using compressive sensing techniques. We present a real time implementation of our algorithm and show its application to an asynchronous time-based image sensor (ATIS) from the Austrian Institute of Technology. The ATIS only provides pixel intensity data when and where a change in pixel intensity is detected, however, noise randomly causes intensity changes to be falsely detected, thereby providing random samples of static regions of the scene. Unlike other compressive sensing imagers, which typically have pseudo-random sampling designed in at extra effort, the ATIS used here provides random samples as a side effect of circuit noise. Here, we describe and analyze a field-programmable gate array implementation of a matching pursuit (MP) algorithm for compressive sensing reconstruction capable of reconstructing over 1.9 million 8×8 pixel regions per second with a sparsity of 11 using a basis dictionary containing 64 elements. In our application to ATIS we achieve throughput of 28 frames per second at a resolution of 304×240 pixels with reconstruction accuracy comparable to that of state of the art algorithms evaluated offline.**

***Index Terms*—Asynchronous, compressive imaging, field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC), FPGA-based real-time video processing.**

I. Introduction

RECENT advances in intelligent CMOS pixel design have spurred a new class of asynchronous time, high dynamic range, high temporal resolution vision sensors [1], [2]. These sensors use an address event representation (AER) [3] scheme in which data output is initiated by the pixels themselves rather than requested by an external source.

For example, in temporal difference (TD) AER sensors [4], [5], pixels only output data when they experience a change in illumination. This means that pixels under constant illumination do not occupy off-chip communication bandwidth, thereby allowing this bandwidth to be used to achieve a higher update rate of pixels which are subjected to changing illumination. The dynamic vision sensor (DVS) [1] from ETH Zurich is one such sensor which detects changes in illumination, although it does not detect the absolute illumination of a pixel.

On the other hand, exposure measurement (EM) AER sensors are designed to measure absolute pixel intensity. Since AER data output is initiated by the pixels themselves, each pixel can integrate for as long or short as it requires to make a measurement. In practice the minimum useful integration time which can be accurately recorded is limited by how quickly an event can be transmitted, while the maximum integration time is limited by dark current and noise. This allows for design of imagers capable of achieving a very high intra-scene dynamic range [6].

The asynchronous time-based image sensor (ATIS) [7] used in this paper (described further in Section II) combines a TD and EM sensor at the focal plane to provide a high dynamic range image sensor which only outputs data when and where changes in intensity are detected. Noise induced false detection of intensity changes provides a random sampling scheme which can be used in a compressive sensing framework.

Compressive sensing allows for reconstruction of missing pixels in incomplete images by assuming the image has a sparse representation in some basis. Taking advantage of this technique can reduce the energy required to acquire an image, since not all pixel intensity values need to be sampled. When implementing compressive sensing, the sampling scheme used must be incoherent with the basis in which the image's sparse representation is to be estimated [8].

Numerous cameras have been custom designed for compressive sensing. The Rice single-pixel camera [9], [10] is a well-known example in which only a single photodetector is used. Light is reflected onto the photodetector using an array of adjustable mirrors. Each illumination measurement from the photodetector therefore represents a summation of the illumination of multiple pixels. Applying compressive sensing techniques to multiple readings acquired sequentially from the camera allows for reconstruction of a 2-D image using fewer samples than would be required by Nyquist sampling. The Rice single-pixel camera has also been proposed for video acquisition, using compressive sensing to estimate a sparse representation of a video sequence [11].

Manuscript received March 02, 2012, revised June 30, 2012; accepted July 22, 2012. Date of publication October 10, 2012; date of current version December 05, 2012. This paper was recommended by Guest Editor G. Setti.

G. Orchard, J. Zhang, Y. Suo, M. Dao, D. T. Nguyen, T. D. Tran, and R. Etienne-Cummings are with the Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: gorchard@jhu.edu; jzhang41@jhu.edu; ysuo1@jhu.edu; minh.dao@jhu.edu; dzungnguyen@jhu.edu; trac.d.tran@jhu.edu@jhu.edu; retienne@jhu.edu).

S. Chin is with the Applied Physics Laboratory, The Johns Hopkins University, Laurel, MD 20723 USA (e-mail: sang.chin@jhuapl.edu).

C. Posch is with the Austrian Institute of Technology—AIT, 1220 Vienna, Austria, and Institut de la Vision/Université Pierre et Marie Curie, 75012 Paris, France (e-mail: christoph.posch@inserm.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2012.2214614

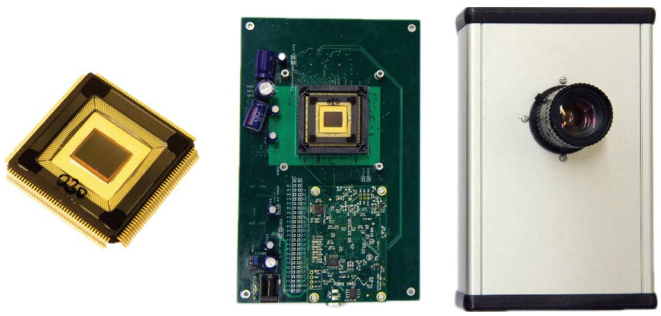


Fig. 1. ATIS chip (left) incorporated into a custom PCB (center) including an opal kelly XEM 3005 FPGA module. Housing and lens mount for the ATIS are shown on the right.

Fig. 2. Typical snapshot showing the outputs of the TD circuit (left) and EM circuit (right) for an outdoor scene. In the TD output on the left, black, or white dots represent pixels which have recently decreased or increased in intensity, respectively. The measurement of pixel intensity shown on the right only occurs when the TD circuit for that pixel has detected a change in intensity.

Robucci *et al.* [12] presented a custom camera which computes the projection of an image onto a noiselet basis set using an analog vector-matrix multiplier at the image plane before digitization to reduce the load on an ADC. They then reconstruct the image by finding the solution with the lowest total variation [13] which matches their measurements.

Many other applications of compressive sensing to compression of conventional video can be found in the literature [14]–[16]. Similar to the camera of Robucci *et al.*, some of these frameworks aim to reduce the burden on ADCs, especially in Terahertz imaging [17], [18]. In the field of medical imaging, compressive sensing has been applied to MRI imaging [19]–[21] to reduce the required signal acquisition time by undersampling k-space, thereby reducing motion artifacts.

The ATIS differs from other front ends used for compressive sensing in that the sampling scheme is random rather than pseudo-random, therefore it is not known until after sampling which locations will be sampled and when. In this paper, we show how compressive sensing can be used to estimate unsampled pixels from the ATIS and how the ATIS can be used as a compressive sensing front end, despite this not being the purpose for which it was designed.

The rest of this paper is structured as follows. We begin by introducing the ATIS camera in Section II and posing the scene reconstruction problem in a compressive sensing framework in Section III. We then discuss possible offline reconstruction algorithms in Section IV before tackling the issue of implementing reconstruction in a real-time system in Section V. The computational requirements of our design are analyzed in Section VI, before we present how parameters for the design were chosen through testing in Section VII. Results from real-time implementation appear in Section VIII, before we conclude with a discussion (Section IX) of this work.

II. Asynchronous Time Image Sensor

ATIS [7] (Fig. 1) from the Austrian Institute of Technology (AIT) is the first AER sensor to contain both an EM sensor and a TD sensor. Furthermore, the ATIS allows these two sensors to be coupled in such a way that a pixel's EM circuit will only initiate a measurement when the TD circuit indicates that a change in intensity has occurred. An example of a typical sensor output is shown in Fig. 2. This means that only pixels which are changing will output their intensity, providing an efficient representation of video, especially in slow changing scenes where most pixels remain constant.

Compression occurs as a direct result of how the data is generated rather than as result of postprocessing to remove redundancy. In practice, noise affects the TD circuit and can randomly trigger false change detection events, even when viewing a static scene under constant illumination. Over time these noise driven measurements will eventually fill the entire scene. This process can take a long time depending on the sensitivity biasing of the sensor which controls how big of a change in intensity must be detected to initiate an intensity measurement. A smaller intensity change threshold would be more likely to be falsely triggered by noise and would therefore result in the scene being filled more quickly.

III. Scene Recovery in Compressive Sensing Framework

Compressed sensing (CS) is a new method of sampling and signal recovery. It predicts that a sparse signal can be exactly recovered from a set of incomplete samples. In order to recover the signal in the CS framework, the signal must have a sparse representation in a particular basis, $\mathbf{\Psi}$, and the samples must be taken randomly in a basis "incoherent" with $\mathbf{\Psi}$. In this section, we briefly discuss the two important notions of "sparsity" and "mutual coherence," and then explain how image recovery from noise-triggered pixel intensity values can be fitted into the CS framework.

A signal is sparse if most of its energy is contained in a few locations of the signal. The total number of these significant coefficients is called signal sparsity, and their locations are denoted as sparsity support locations. A good example of a sparse signal is a sinusoidal wave containing a single frequency, as it can be represented by a Dirac delta function at a single location in a frequency basis.

Natural images are also good examples of sparse signals when represented in their frequency domain because most of the signal energy is concentrated at a few low frequencies. In fact, many image compression techniques, such as JPEG, have taken advantage of this phenomenon to achieve a remarkable rate of image compression. The JPEG compression algorithm first breaks an image up into 8×8 pixel patches, and then transforms each patch into its frequency representation using two-dimensional discrete cosine transform (2D-DCT). Since

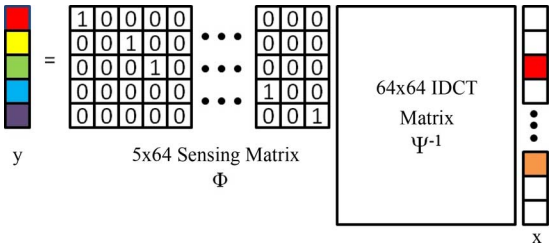


Fig. 3. Graphic illustration of the construct of $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{\Psi}^{-1}\boldsymbol{x}$.

most of the energy is represented by only a few frequency coefficients, a lot of the other small coefficients could be discarded without causing much distortion to the image.

The notion of "Mutual Coherence" [22] is a measure of the greatest correlation between elements belonging to two different orthonormal bases in R^N. It is defined mathematically as

$$\begin{array}{rcl} \mu(\boldsymbol{\Phi},\boldsymbol{\Psi}) & = & \sqrt{n}\max_{1\le k,j\le n}|\langle\boldsymbol{\phi}_j,\boldsymbol{\psi}_k\rangle| \\ \mu(\boldsymbol{\Phi},\boldsymbol{\Psi}) & \in & [1,\sqrt{n}], \text{ and } \boldsymbol{\Phi},\ \boldsymbol{\Psi}\in R^{N\times N} \end{array} \quad (1)$$

where μ is the mutual coherence, $\boldsymbol{\phi}_j$, $\boldsymbol{\psi}_k$ are elements in $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, respectively, which are two orthonormal bases of R^N. If $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ contain correlated elements, their mutual coherence will be big, otherwise it will be small. If two bases have a small mutual coherence, then it can be shown that a sparse signal represented in one base would not be sparse in the other base. A good example is that a sinusoid is sparse in the frequency domain but dense in the spatial domain because these two domains have a minimal mutual coherence of $\mu = 1$. Therefore, if we sample randomly at sparse signal's incoherent domain, there will be high probability that we acquire most of the signal information. Therefore, for compressed sensing, we desire to sample the signal randomly in its incoherent domain.

The noise triggered pixel intensity samples from the ATIS can be treated as random samples of the image in the spatial domain. Thus, the compressed sensing method could be used conveniently to recover the entire image from these samples. The recovery process is to solve an optimization problem to find an estimate signal that has the smallest L1-norm [23], defined as

$$\hat{\boldsymbol{x}} = \arg\min\|\boldsymbol{x}\|_1 \text{ subject to } \boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{\Psi}^{-1}\boldsymbol{x} \quad (2)$$

where $\hat{\boldsymbol{x}} \in R^N$ is the estimation of the image $\boldsymbol{x}$, while $\boldsymbol{y} \in R^M$, is given by the noise triggered samples of the ATIS; $\boldsymbol{\Psi}^{-1} \in R^{N\times N}$ is the inverse discrete cosine transform basis matrix; $\boldsymbol{\Phi} \in R^{M\times N}$ is an $M \times N$ matrix with rows selected from an identity matrix corresponding to the location of the samples. The example below shows how we could construct $\boldsymbol{\Phi}$ from the noise triggered pixel intensity value y.

Let us assume we are trying to recover an 8×8 image $\boldsymbol{x}$. At the sensor side, we record noise triggered pixel values, $\boldsymbol{y}$, at locations 1, 3, 4, 63, and 64 of the image. We could then construct sampling matrix $\boldsymbol{\Phi}$ containing the 1st, 3rd, 4th, 63rd, and 64th rows of a 64×64 identity matrix. Fig. 3 gives a graphic illustration of the construct of $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{\Psi}^{-1}\boldsymbol{x}$.

The L1-minimization problem presented above can be solved by convex optimization method such as basis pursuit [24], or by a group of greedy methods such as matching pursuit (MP) [25], orthogonal matching pursuit (OMP) [26] and subspace pursuit (SMP) [27]. Greedy methods are generally preferred for hardware implementation due to their simplicity and speed. Compared to other types of greedy pursuits, MP offers the best trade-off between computation complexity and reconstruction quality, therefore we choose MP as our recovery algorithm.

A. Matching Pursuit

The MP algorithm is a well known, well defined algorithm, the details which can be found in the references [25]. Nevertheless, a brief explanation is included here for completeness. To better understand the algorithm let us make the following definitions.

- $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}^{-1}$.
- $\boldsymbol{A_\lambda} \in R^M$ is the λth column of $\boldsymbol{A}$.
- $\boldsymbol{A_s} \in R^{M\times S}$ is the MP estimation matrix, where S corresponds to the predefined signal sparsity, which will also be the number of MP iterations performed.
- $\boldsymbol{y} \in R^M$ contains the measured intensity values from ATIS.
- $\boldsymbol{y}_p \in R^M$ is an estimate of $\boldsymbol{y}$.
- $\boldsymbol{y}_r \in R^M$ is the residual error in the estimate of $\boldsymbol{y}$.
- $\hat{\boldsymbol{x}} \in R^N$ is an estimate of $\boldsymbol{x}$.
- $\hat{x}_k \in R$ is the kth element of $\hat{\boldsymbol{x}}$.

The steps of the MP algorithm can be summarized as follows.

1) Initialize $\boldsymbol{A_s} = \vec{0}; \boldsymbol{y}_r = \boldsymbol{y}; k = 0$.
2) Increment k.
3) Perform the dot product of $\boldsymbol{y}_r$ with every column of $\boldsymbol{A}$ and determine the index, λ, of the column that corresponds to the greatest dot product value.
4) Perform least squares estimate to calculate $\hat{x}_k$ using $\boldsymbol{A_\lambda}$. In other words solve

$$\hat{x}_k = (\boldsymbol{A}_{\boldsymbol{\lambda}}^T\boldsymbol{A_\lambda})^{-1}\boldsymbol{A}_{\boldsymbol{\lambda}}^T\boldsymbol{y}_r. \quad (3)$$

5) Project $\hat{\boldsymbol{x}}_k$ onto $\boldsymbol{A}_\lambda$ to calculate $\boldsymbol{y}_p$

$$\boldsymbol{y}_p = \boldsymbol{A}_\lambda\hat{x}_k. \quad (4)$$

6) Update the residual signal, $\boldsymbol{y}_r$, for the next iteration

$$\boldsymbol{y}_r \leftarrow \boldsymbol{y}_r - \boldsymbol{y}_p. \quad (5)$$

7) Augment $\boldsymbol{A_s}$, with $\boldsymbol{A_\lambda}$. Augment $\hat{x}$ with $\hat{x}_k$.
8) Repeat steps 2 to 7 until $k = S$ (the number of iterations reaches the number of sparsity, S). $\hat{\boldsymbol{x}}$ will be the sparse signal represented in $\boldsymbol{\Psi}$ domain. The estimate of $\boldsymbol{y}$ can be obtained using

$$\hat{\boldsymbol{y}} = \boldsymbol{A}_s\hat{\boldsymbol{x}}. \quad (6)$$

IV. Offline Reconstruction

Real-time reconstruction algorithms are constrained by the time available for computation. Here, we investigate offline reconstruction approaches for comparison with our real-time implementation. Two different offline approaches based on the

concept of low-rank matrix completion and low-rank tensor approximation are presented.

In the low-rank matrix completion approach, similar patches are grouped in both temporal and spatial domains to construct a low-rank structure. The matched blocks are converted into vectors and stacked into a low-rank matrix $\mathbf{D}$. The first column of $\mathbf{D}$ is the block to be reconstructed in vector form and the remaining columns are the matching blocks. Intuitively, the underlying structure of all columns of D should be similar thus the matrix $\mathbf{D}$ becomes an incomplete low-rank matrix with missing values. Generally, $\mathbf{D}$ can be represented as

$$\mathbf{D} = [\mathbf{A} + \mathbf{Z}]_{\Omega} \tag{7}$$

where $\mathbf{A}$ is a low-rank matrix, $\mathbf{Z}$ is the sparse noise that comes from the object motion, and Ω is the index set of reliable pixels. The reconstruction of $\mathbf{A}$ from its incomplete observation $\mathbf{D}$ is the robust principle component analysis (RPCA) [28] problem and can be formulated as follows:

$$\begin{aligned} min_{\mathbf{A},\mathbf{Z}} \quad & \|\mathbf{A}\|_* + \lambda\|\mathbf{Z}\|_1 \\ s.t. \quad & [\mathbf{A} + \mathbf{Z}]_{\Omega} = \mathbf{D} \end{aligned} \tag{8}$$

where $\| \ \|_*$ denotes the nuclear norm of a matrix, $\| \ \|_1$ denotes the sum of the absolute values of matrix entries, and λ is a positive weighting parameter. The augmented Lagrange multiplication algorithm [29] is used to solve the minimization problem (8). Different from the low-rank matrix completion approach, the low-rank tensor approximation approach uses similar patches to construct a 3-D tensor rather than a 2-D matrix [30]. Laplacian interpolation is used to fill in the missing data for initialization. Taking into account both the temporal and spatial correlation, each patch with missing values is grouped with similar 2-D patches according to block matching criteria and stacked into a third order tensor. Assuming that the constructed tensor has a low-rank structure in the third dimension, the algorithm uses iterative Tucker decomposition [31] to approximate the desired tensor with small mode-3 rank. After the iterative process, the patch being recovered will have a refined estimate of the values at the missing locations. Generally, each patch with missing data is filled independently and in an order from small missing ratio to large missing ratio.

For the sake of completeness, we also compare the performance of field-programmable gate array (FPGA) implementation with an offline nonparametric Bayesian approach which uses beta process as prior for dictionary learning [32]. For all approaches, a patch size of 8×8 is used and in each iteration patches are shifted by one pixel in horizontal, vertical or diagonal direction to achieve better result.

Results from the tensor, RPCA, and BPFA algorithms are shown in Fig. 10 and discussed in Section VIII-B alongside real-time results from our FPGA implementation.

V. Real-Time Hardware Implementation

A. When and Where to Reconstruct

When implementing reconstruction in hardware we must determine when and where to perform reconstruction since asynchronous imagers do not have a notion of a "frame," so data can arrive from any location at any time. Three obvious approaches could be taken. The first is to reconstruct when and where new data is available. The second is to reconstruct at fixed time intervals, but only where new data is available, and the third is to reconstruct the entire scene at fixed intervals.

In terms of the required computation, the first approach would be best if very little new data is available. For example, if only one new pixel intensity measurement is made in a fixed time period, then only the 64 overlapping patches of which that pixel is a member need be reconstructed in that time period. However, if the pixel in question provides multiple intensity updates in a fixed time period, then the reconstruction of the 64 overlapping patches containing that pixel would have to be recomputed multiple times in that time period.

When a pixel is frequently updating its intensity, the second approach of reconstructing at fixed time intervals (but only where pixel intensities have updated), places a limit on the maximum number of times reconstruction of the 64 overlapping patches containing that pixel will be performed within a fixed time period. The second approach therefore requires less computation than the first approach when parts of the scene are rapidly updating.

The third approach is similar to the second, in that reconstruction is performed only at fixed time intervals, thereby limiting the maximum computation required in a fixed time period. The difference between the second and third approaches is that the third approach reconstructs everywhere, whereas the second only reconstructs where new data is available. The second approach is therefore clearly more computationally efficient, but it also requires far more storage because the results of each reconstructed 8×8 patch must be individually stored to allow for the combination of recently reconstructed patches with previously reconstructed patches which contain no new pixel intensity measurements. The memory requirement for storing these reconstructed patches for the second approach is far beyond (more than 10x) what is available on the targeted FPGA. However, this memory requirement is overcome in the third approach by reconstructing patches in a predefined sequence which allows for them to be combined on the fly as individual patch reconstruction results become available.

The third approach is the one taken in this work. Although not the most computationally efficient, the memory requirements of the first two approaches render them unrealizable on the targeted FPGA. Finally, unlike the first approach, the third approach allows us to predict the required computation and therefore guarantee throughput in frames per second (FPS). If enough memory were available, the second approach could be used and would be guaranteed to have equal or greater throughput.

B. VHDL Code Structure

The hardware setup for our real-time implementation of compressive sensing scene reconstruction is shown in Fig. 4. The MP algorithm described in Section III-A was implemented in VHDL code to run on a Xilinx Virtex 6 ML605 development board containing a XC6VLX240T FPGA. Raw AER data from the ATIS is streamed over gigabit Ethernet to the Xilinx Virtex 6 evaluation board (part number ML605 available for USD$1795). Reconstruction is performed on the FPGA and

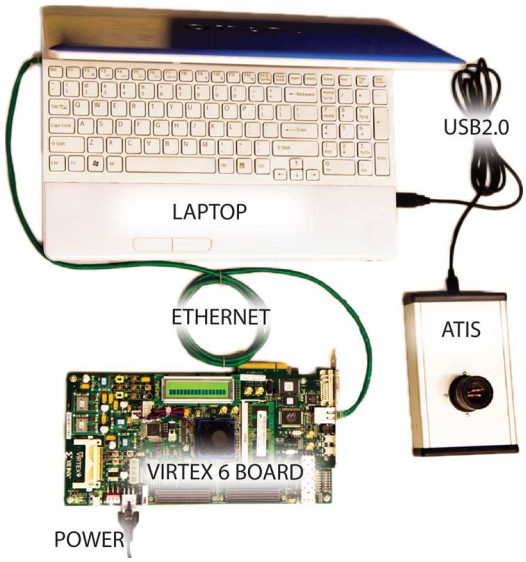


Fig. 4. Hardware setup for real-time compressive sensing scene reconstruction using the ATIS. AER data from the ATIS is transmitted to the host PC via a USB2.0 interface. The data is then forwarded (unprocessed) to the FPGA board via ethernet. The FPGA interprets the incoming AER data, performs reconstruction, scales the results and transmits them back to the host PC (see Fig. 5).

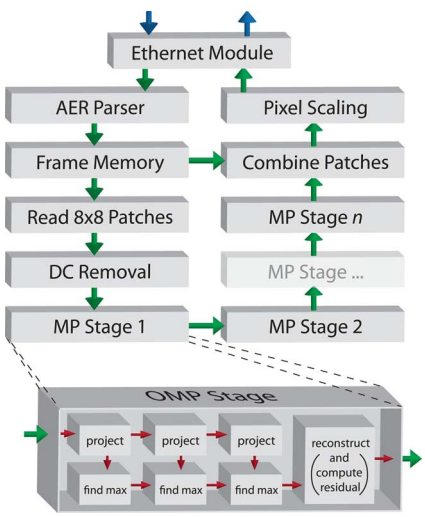


Fig. 5. Block diagram of VHDL code architecture. Each module is capable of a throughput of 64 pixels (1 patch) every 64 clock cycles, resulting in a non-blocking pipelined architecture. The architecture is reconfigurable in that the number of MP stages can be changed, and the size of the basis used by each stage can be changed (see Section V-D2). The pullout at the bottom shows an example configuration of the submodules contained in each MP stage.

reconstructed pixel intensity values are returned over Ethernet to the C++ GUI for display.

The structure of the VHDL code implemented on the FPGA is shown in Fig. 5. The *AER Parser* module converts incoming AER data into 16 bit pixel intensity values (Section V-C), which are used to update a representation of the frame in internal RAM (the *Frame Memory* module). A separate module reads patches of size 8×8 pixels (with a shift of 1 pixel between subsequent patches) and sends them to the *dc Removal* module which removes any dc offset. The zero offset patches then pass through a series of identical *MP Stages* (Section V-D2), each of which computes the value of a single basis coefficient. The results of the MP reconstructions are passed to the *Combine Patches* module which combines the patches (Section V-D3) and replaces estimated pixels with known pixel intensity values from *Frame Memory* wherever they are available. Finally, the *Pixel Scaling* module (Section V-E) converts the high dynamic range (16 bit) pixel intensity values to an 8 bit range for transmission back to the host PC where they will be displayed.

C. AER Parser

Measured pixel intensity values are inferred from AER data by measuring the time difference between subsequent events from a pixel. An internal RAM module holds the timestamp of the first event from each pixel. When a second event is received from a pixel, its timestamp is subtracted from the timestamp of the first event, giving a time difference which represents the pixel value. These pixel intensity values are stored in internal RAM by the *Frame Memory* module in Fig. 5. The *Frame Memory* module allows for pixel intensity values to be read and written simultaneously, meaning that a pixel intensity value can be overwritten with a new value as soon as it is available, regardless of whether the reconstruction module is also accessing the RAM.

D. Reconstruction

Frame patches measuring 8×8 pixels are read at 1 pixel per clock cycle, with a shift of 1 pixel between subsequent patches. Reconstruction is performed on data in a streaming fashion as it enters each module, resulting in a fully pipelined architecture which can start receiving a new patch as soon as it has finished receiving the previous patch. This results in a sustained throughput of one 8×8 pixel patch every 64 clock cycles.

1) DC Removal: The first stage of reconstruction removes dc offset (the mean of the known pixels) before MP reconstruction. Due to the $1/f$ power spectrum which is characteristic of natural scenes [33], the dc coefficient is likely to more accurately represent a frame patch than any other single DCT coefficient. The dc removal stage could be omitted and replaced with another MP stage which contains a dc basis element, but our dc offset removal stage is far faster and less resource intensive than an MP stage. The dc removal stage can be seen as removing background illumination, leaving the MP stages to represent only the differences in pixel illumination.

2) MP Module: Each MP stage consists of three types of submodules as shown in Fig. 5. Each *project* submodule computes the projection of the input onto a set of 64 basis elements, while each *find max* submodule determines which of these projections was the largest. Multiple *project* and *find max* submodules can be cascaded within a single stage as shown in Fig. 5, with each *project* submodule projecting the input onto a different basis. Pipelining in this manner allows reconfiguration to use multiple bases for reconstruction without hindering throughput. Once the best basis element has been chosen, the *reconstruct* submodule scales that basis element so as to minimize the residual error in estimation. The residual error and current estimate of the signal are then passed to the next stage.

3) Recombination: For each pixel location there will be 64 estimates of its value, one from each of the 64 overlapping patches which contains the pixel location. The recombination

module simply computes the mean of all these estimates. If we already have a reading for the pixel location in question then we swap out the estimated pixel value with the existing measurement. Patches which overlap the edges of the frame are not reconstructed, therefore pixels near the edge will have fewer than 64 estimates.

E. Scaling

The *Pixel Scaling* module converts the 16 bit pixel intensity values to 8 bit values using (9) before they are returned to the host PC over gigabit Ethernet. The parameters *offset* and *scaling* are determined by the host PC from AER data captured when the camera initializes

$$\begin{aligned} \text{pixel}_{\text{scaled}} &= \text{offset} + \frac{\text{scaling}}{\text{pixel}_{HDR}} \\ \text{offset} &= \frac{255}{\left(1 - \frac{P_{\max}}{P_{\min}}\right)} \\ \text{scaling} &= \frac{255}{\left(\frac{1}{P_{\min}} - \frac{1}{P_{\max}}\right)} \end{aligned} \tag{9}$$

where $P_{\min}$ and $P_{\max}$ are the smallest and largest measured pixel intensity values, respectively.

VI. Computational Requirements

A. Multiply Accumulates

The computational requirements to implement reconstruction are outlined below in terms of multiply accumulate operations per second (MACS)

$$\text{MACS}_{\text{total}} = \text{MAC}_{\text{frame}} \times FPS \tag{10}$$

where $\text{MACS}_{\text{total}}$ is the number of MACS required to achieve the desired frames per second (FPS) and $\text{MAC}_{\text{frame}}$ is the number of multiply acculumate (MAC) operations required to reconstruct a single frame

$$\text{MAC}_{\text{frame}} = k \times \text{MAC}_{\text{MP stage}} \times P_{\#} \tag{11}$$

where k is the number of MP stages to be implemented, $P_{\#}$ is the number of patches requiring reconstruction per frame, and $\text{MAC}_{\text{MP stage}}$ is the number of MAC operations required per MP stage. $P_{\#}$ can be calculated from

$$P_{\#} = (I_{\text{width}} - P_{\text{width}} + 1) \times (I_{\text{height}} - P_{\text{height}} + 1) \tag{12}$$

where I_{width} and I_{height} are the frame width and height respectively in pixels. P_{width} and P_{height} are the width and height, respectively, of reconstruction patches in pixels

$$\begin{aligned} \text{MAC}_{MP\ stage} = &project_{\#} \\ &\times (\text{MAC}_{\text{project}} + \text{MAC}_{\text{findmax}}) \\ &+ \text{MAC}_{\text{reconstruct}} \end{aligned} \tag{13}$$

where $\text{MAC}_{\text{project}}$, $\text{MAC}_{\text{findmax}}$, and $\text{MAC}_{\text{reconstruct}}$ are the number of MAC operations required by each *project*, *find max*, and *reconstruct* submodule, respectively. $project_{\#}$ is the number of *project* submodules per MP stage. In practice the $\text{MAC}_{\text{findmax}}$ submodule does not require any multiplications. The MAC requirements for the $\text{MAC}_{\text{project}}$ and $\text{MAC}_{\text{reconstruct}}$ submodules are given by

$$\begin{aligned} \text{MAC}_{\text{project}} &= P_{\text{width}} \times P_{\text{height}} \times D_{\text{length}} \\ \text{MAC}_{\text{reconstruct}} &= 3 \times P_{\text{width}} \times P_{\text{height}} + 1 \end{aligned} \tag{14}$$

where D_{length} is the number of dictionary elements per *project* submodule.

B. Pipelining

Using (12), a single full frame reconstruction at 240×304 pixel resolution (the resolution of the ATIS) using 8×8 patches contains 69201 patches. If the reconstruction module can receive an $8 \times 8 = 64$ pixel patch every 64 clock cycles, then the system can reconstruct an entire frame every

$$64 \times 69201 = 4.4 \times 10^6 \ \text{ clock cycles.} \tag{15}$$

To achieve a real-time frame rate of 24 FPS, we would require

$$24 \times 4.4 \times 10^6 = 106 \times 10^6 \ \frac{\text{clock cycles}}{\text{second}}. \tag{16}$$

This means the design must run at a clock frequency of over 106 MHz to maintain a frame rate of 24 FPS.

VII. Parameter Selection

Matlab code was written to exactly mimic the VHDL code for use as a simulation tool to help determine parameters to be used. This Matlab simulation was used to investigate the effect of patch size, the basis dictionary, and the number of sparsity stages used. Initially a 2D-DCT dictionary was used to determine patch size. Details of the dictionary can be found in Section VII-B. Matlab code was used instead of running the VHDL code itself on the FPGA to allow quick reconfiguration of parameters and access to high dynamic range reconstruction results for peak signal-to-noise ratio (PSNR) calculation as a metric of reconstruction accuracy (in VHDL the HDR reconstruction results are scaled to 8 bits by the *Pixel Scaling* stage from Fig. 5 before being transmitted to the PC). PSNR was used as the metric for reconstruction accuracy because it is the standard criteria used for image quality assessment [34]. The equation used to compute PSNR is shown below

$$PSNR = 10 \times \log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{17}$$

where MAX is the maximum pixel value in the scene and MSE is the mean square error. Due to the high dynamic range of data provided by the ATIS, MAX can vary greatly between scenes and is therefore calculated separately for each scene.

Plots shown represent the mean result from simulations run on 10 different scenes captured with the ATIS. These scenes are shown in Fig. 6. Each scene is simulated with the percentage

Fig. 6. The ten different images captured by the ATIS for use in parameter selection. The images are a combination of indoor and outdoor images with different focal lengths to capture features at various scales.

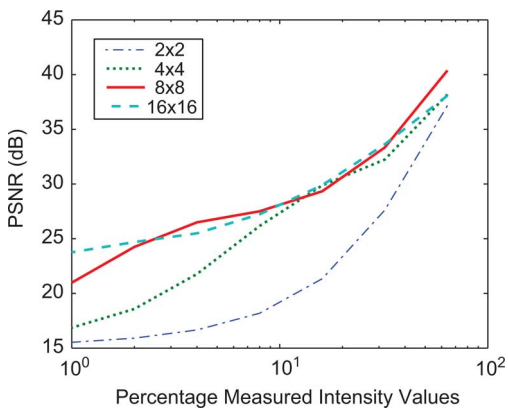


Fig. 7. PSNR as a function of patch size at different percentages of measured pixel values for a 64 element 2D-DCT basis and 11 MP stages. When a low percentage of pixel intensity values are known, the PSNR depends on the patch size, with large patch sizes producing better results because they can cover the entire scene with estimates when less pixel intensity values are known. As more pixel intensity values are measured, the 8×8 patch size performs comparably to 16×16 patches.

of measured pixel intensity values beginning at 1% and doubling between simulations until 64% of pixel intensity values are known.

A. Patch Size

Fig. 7 shows the results when using 10 MP stages with a complete 2D-DCT basis at patch sizes of 2×2, 4×4, 8×8, and 16×16. PSNR when only a few pixel intensity values are known is determined by how much of the scene can be covered with estimates, giving larger patches an advantage. At 2% known pixel intensities, the 8×8 patch size is already comparable to 16×16 and eventually surpasses it at higher percentages of measured pixel intensities (64%). Presumably this increase is because the 8×8 patch makes better use of local information to estimate pixel intensity values. The 4×4 patch

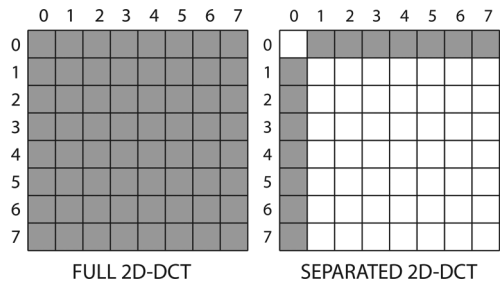


Fig. 8. Locations of 2D-DCT coefficients (indicated by shaded regions) used to create the 2D-DCT dictionary (left) and the smaller *separated* 2D-DCT dictionary (right).

size requires more measured pixel intensity values to cover the scene, but produces comparable results once the scene is covered with estimates. 2×2 patches consistently under perform against the other sizes because even once the scene is covered with estimates, a 2×2 patch does not contain enough information to estimate frequency content.

We chose a patch size of 8×8 because it is large enough to give good accuracy at a low percentage of known pixel intensities, while still being small enough to be easily implemented in real time, and it is the size used by the JPEG standard for compression. By comparison, a 16×16 patch size would require a 4 times larger dictionary, with each dictionary element being 4 times larger. Therefore, 16 times more computational resources are required to compute projections onto the dictionary in real-time.

B. Dictionary

The 64 elements of the full 2D-DCT dictionary were created by taking the 2D inverse discrete cosine transform (2D-IDCT) of 64 different matrices, each with a "1" in a single location and zeros elsewhere. Shaded regions on the left side of Fig. 8 show all possible 2D-DCT coefficient locations (locations for a "1" before applying multiply by the 2D-IDCT matrix).

A second, much smaller, *separated* 2D-DCT dictionary was generated for testing purposes. This *separated* dictionary was created by limiting the dictionary such that each elements varies along only the horizontal or only the vertical direction. For the *separated* dictionary, possible locations for a "1" in the 2D-DCT matrix before multiplying by the 2D-IDCT matrix are shown as the shaded regions on the right of Fig. 8. The dc element (location $[0, 0]$) was omitted because a dc removal stage already exists in the VHDL architecture. In other words, the *separated* 2D-DCT dictionary consists only of vertically and horizontally oriented gratings of different frequencies.

For both 2D-DCT dictionaries each element was normalized to have an L2-norm of 255. A 2D HAAR wavelet dictionary was created in a similar fashion.

After testing, the 2D-DCT basis was chosen. Our architecture allows for the use of multiple bases, but testing showed that using a HAAR wavelet basis produced worse results than 2D-DCT across all missing ratios and all 10 scenes tested. Using both the HAAR and 2D-DCT bases produced better results than the HAAR basis by itself, but still performed worse than when only the 2D-DCT basis was used. 2D-DCT coefficient are also the standard used by JPEG compression

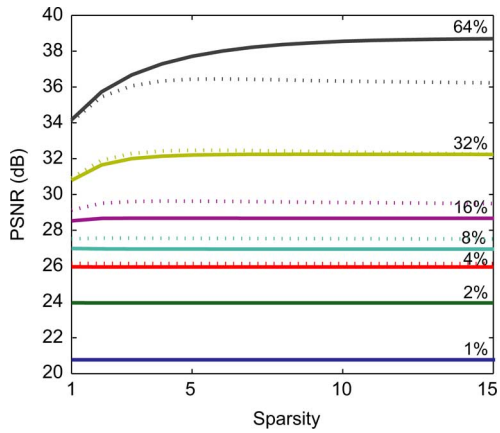


Fig. 9. Reconstruction accuracy as a function of the number of MP stages (sparsity) for different percentages of measured pixel intensities (listed above each plot line) using an 8×8 pixel patch size. Solid lines indicate PSNR resulting from a full 2D-DCT dictionary, while dotted lines indicate PSNR when the dictionary contains only 1D-DCT elements in the x and y directions. Each data point is the mean result for 10 different scenes. coefficients.

C. Sparsity

Each *project* submodule (see Fig. 5) requires 64 multipliers (DSP48E1 blocks), one per basis element onto which a projection must be computed, while each *reconstruct* module requires 4 multipliers. Since only the 2D-DCT basis is being used (1 *project* submodule), each MP stage requires 68 multipliers. The targeted Virtex 6 FPGA has 768 DSP48E1 blocks, allowing for a maximum of 11 MP stages.

Fig. 9 shows how the number of MP stages affects reconstruction accuracy when different percentages of pixels are known. Each MP stage estimates a single basis coefficient, therefore the number of MP stages is equal to the number of nonzero basis elements used for reconstruction (i.e., the sparsity of the reconstruction estimate in the basis space).

As expected, when more pixel intensity values are known, the PSNR is higher. When only a few pixel intensity values are known, a higher sparsity does not significantly improve PSNR, since there is not sufficient data to estimate a high number of coefficients. The more pixel intensity values available, the more a high sparsity increases PSNR, as can be seen from plots at 32% and 64% known pixel intensities. Above a sparsity of 10, not much change is seen in PSNR.

In Fig. 9, the dotted lines show results when the *separate* dictionary is used. This *separated* dictionary performs as well as the full 2D-DCT dictionary up to 2% known pixel intensities, above which it performs marginally better. However, at 64% the *separated* dictionary performs significantly worse than the full dictionary.

A sparsity of 11 using the full 2D-DCT dictionary was used in our design because this is the highest sparsity we can implement with the full dictionary on the FPGA (limited by resources) and because higher sparsity helps reconstruction at high percentage of known pixel intensities, without significantly hurting reconstruction at low percentages of known pixel intensities.

TABLE I
FPGA RESOURCES USED AND LATENCY OF EACH MODULE

Module	Slices	Multipliers (DSP48E1)	Block RAM	Latency (clock cycles)
AER Parser	308	0	165	4
DC Removal	198	0	0	128
MP Stage	583	68	9	388
project	278	64	8	64
find max	18	0	0	4
reconstruct	286	4	1	128
Recombination	488	0	3	32
Scaling	688	0	0	32
Total	9126	748	295	4.4×10^6
Available	37680	768	416	-

VIII. RESULTS

The VHDL code was implemented using 11 MP stages with a 64 element 2D-DCT basis, operating on patch sizes measuring 8×8 pixels each. Modelsim was used to simulate the code and confirm that results exactly match the Matlab simulations presented in Section VII. Analysis of simulations and results from synthesis are presented in Section VII-A below. The VHDL code was then implemented on the targeted Virtex 6 FPGA and used for real-time reconstruction as illustrated in Fig. 4. This setup was used as a final test to verify the speed and accuracy of the system, results of which are shown in Section VII-B.

A. FPGA Speed and Resources

Analysis of the FPGA code using Modelsim and the Xilinx ISE showed that the design is capable of operating at a clock speed of 125 MHz, which translates into a frame rate of $125/4.4 = 28$ FPS. Power consumption of the Virtex 6 FPGA was estimated at 4.8 W using the Xilinx XPower Analyzer tool. Table I shows the resources used by the design and the latency added by each module. The *MP Stage* row in Table I reflects the total resources required per MP stage in the design. These values are equal to the sum of the resources required for the *project*, *find max*, and *reconstruct* substages. The *Total* row represents the total resources used by the design, including the Ethernet interface. The latency figure given for the *Total* row is the time from when the first pixel is read, to the time when the last reconstruction result is completed.

B. Reconstruction Accuracy

Fig. 4 shows the setup into which the reconstruction was incorporated. As predicted by simulation, the system ran reliably at a clock frequency of 125 MHz and a frame rate of 28 FPS was verified by observing incoming Ethernet packets using WireShark on the host PC. Test scenes from simulation (Section VII) were reconstructed using the FPGA and the results were compared to simulation results to verify that simulations were accurate.

Fig. 10 quantifies the accuracy of our MP implementation (using both the full and separated 2D-DCT dictionaries) and compares it against the algorithms discussed in Section IV, for which reconstruction was computed offline. When freed from the constraint of operating in real-time, tensor approximation consistently provides the best results across all percentages of known pixel intensities. RPCA is always within 1 dB PSNR of the tensor approximation, while MP with 11 stages is always within 2 dB (for both the separated and full 2D-DCT basis). BPFA performs comparably when more than 16% of

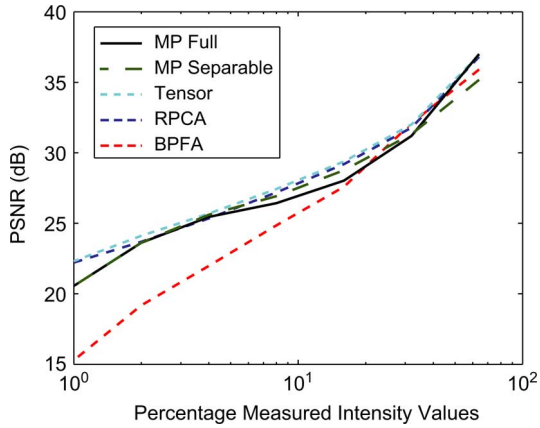


Fig. 10. Comparison of different reconstruction methods. Offline tensor approximation provides the highest reconstruction accuracy, but both the full and separable real-time matching pursuit (MP) implementations are within 2 dB PSNR across all percentages of known pixel intensities. Robust principal component analysis (RPCA) and beta process factor analysis (BPFA) results are also shown for reference.

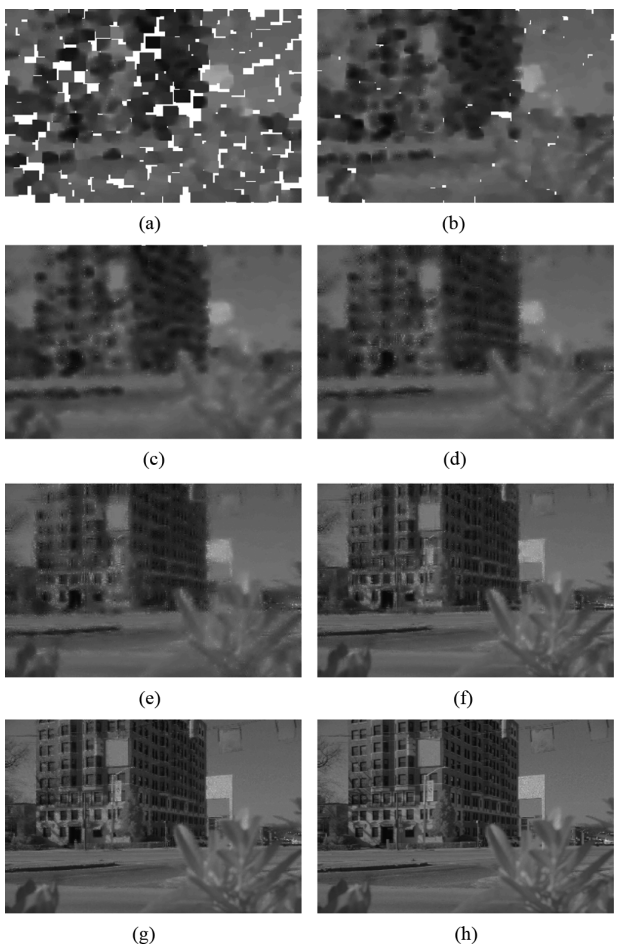


Fig. 11. An example of the output from reconstruction of an outdoor scene captured with the ATIS at (a) 1%, (b) 2%, (c) 4%, (d) 8%, (e) 16%, (f) 32%, (g) 64%, and (f) 100% known pixel intensities when using a sparsity of 11 and a 2D-DCT basis.

pixel intensity values are known, but PSNR degrades significantly below 16% known pixel intensities.

Fig. 11 shows example results from the system in operation. At 1% and 2% known pixel intensities, the 8×8 patches are not large enough to cover the entire scene, resulting in a low PSNR as reflected by Fig. 9. By 4% the entire scene is filled, but fine details still appear blurry. At 16% and 32% different contents of the scene become easily recognizable, but the scene still appears distorted. By 64% this distortion has disappeared.

C. Extension to Higher Resolution

Although we opted to use the full 2D-DCT dictionary, a *separated* dictionary containing only 1D-DCT elements in the x and y directions should be kept in mind if attempting to reconstruct at higher resolutions. The *separated* dictionary significantly reduces computational requirements by reducing D_{length} in (14) from 64 to 14. Furthermore, since the 1D-DCT elements contain only 8 different coefficients rather than 64, the number of multiplications required can be reduced even further. Taking advantage of this, (14) can be altered to become

$$\begin{aligned} \text{MAC}_{project\ separable} &= P_{\text{width}} \times D_{\text{length}} \\ \text{MAC}_{reconstruct\ separable} &= 3 \times P_{\text{width}} + 1 \end{aligned} \tag{18}$$

assuming square patches are used ($P_{\text{width}} = P_{\text{height}}$).

Using this *separated* dictionary will affect reconstruction accuracy, as seen in Fig. 9, but will allow for faster reconstruction (or use of less computational resources) as shown in Fig. 12. The implementation presented in this paper has 11 MP stages and runs at 28FPS, resulting in an $MP \times sparsity$ product of 308, which is predicted well by Fig. 12 where the *XC6VlX240T* and *ATIS* lines intersect.

Fig. 12 shows that reconstruction with a full 2D-DCT dictionary at full HD resolution (1920×1080) would not be possible even with a top of the range Virtex 7 (XC7VX980T). However, using the *separated* dictionary full HD could be reconstructed at 60 FPS with 5 MP stages on the same Virtex 6 as our model. Fig. 9 shows that using more than 5 MP stages does not provide any advantage for the *separated* dictionary.

Comparison with other computing platforms in Fig. 12 shows that higher performance can be achieved using a CPU or GPU. In practice the performance of such platforms is often limited by memory bandwidth, not MACS. The MACS reported for FPGAs all assume a conservative 125 MHz clock, but fully optimized designs can run at up to 600 MHz, providing over 4x higher performance than reported in Fig. 12.

IX. Discussion

This paper shows how a sensor can be optimized to deliver the specific type of data that allows random spatiotemporally sparse sampling of images, streaming of asynchronously generated pixel data, low-bandwidth communication from the sensor to the reconstruction infrastructure, and real-time reconstruction of video using compressed sensing approaches. The current sensor is relatively small in terms of today's multi-megapixel cameras, however, is possible to fabricate a megapixel version of the ATIS using the current pixel layout within the same chip area as a digital single lens reflex (DSLR) image sensor. Even higher resolution can be achieved with minor redesign of the pixel circuitry, e.g., to optimize for pixel size rather than image

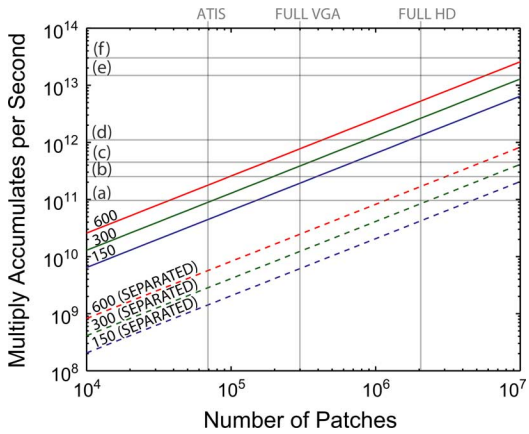


Fig. 12. Computational requirements for reconstruction using 8×8 patches, in terms of MACS (vertical axis). The number of patches per frame is shown on the horizontal axis and can be computed using (12) for different resolutions. The values for the ATIS and common video resolutions are shown for reference. Plot lines represent different $FPS \times sparsity$ values as marked. For example, $FPS \times sparsity = 300$ could be a sparsity of 10 at 30 FPS or a sparsity of 5 at 60 FPS. Dotted lines indicate the use of the *separated* DCT basis. Horizontal lines indicate the computational capacities of various platforms (a) XC6VlX240T (FPGA used in our implementation), (b) XC6VSX475T (top of the range Virtex 6 FPGA), (c) XC7VX980T (top of the range Virtex 7 FPGA), (d) Intel Core i7 980 Extreme CPU (double precision operations per second), (e) Nvidia Tesla C2050 GPU (single precision operations per second), (f) AMD FireStream 9270 GPU (single precision operations per second).

dynamic range, while using a more aggressive integrated circuits fabrication node, e.g., 45 nm or smaller as outlined in the International Technology Roadmap for Semiconductors [35] rather than the currently used 180 nm technology. Such developments will make compressed sensing based video acquisition ideal for mobile applications with limited communications bandwidth. The reconstruction approaches presented in this paper show that both online, real-time and offline, high accuracy reconstruction are possible using this specialized compressed sensing video front-end.

Furthermore, the ATIS sensor produces data in a manner similar to human retinae, while the real-time FPGA reconstruction of images from asynchronous streams of ATIS pixel intensity values can be seen to mimic the cortical processes that allows a person to recognize objects after only a small percentage of neural spikes from the retinae has been received by the brain. A natural extension of this research is to add a neuromorphic object recognition engine as part of the reconstruction, which will present new questions at the interface of compressed sensing video reconstruction and machine learning. Developments in this area will increase the similarities between the work presented in this paper and biological visual information processing and understanding.

Design of the implementation described in this paper was guided by the resources available on the targeted FPGA and the resolution of the ATIS, but the modular nature of the VHDL code allows it to be easily reconfigured for other applications. The implementation presented here uses 11 MP stages because the resources to do so are available on the FPGA, but Fig. 9 suggests that using fewer stages would not significantly affect accuracy. A similar implementation with fewer MP stages (which therefore uses fewer resources), could be implemented if FPGA resources are required for other tasks, such as object recognition or motion estimation. Further size and speed improvements can be realized by optimizing the VHDL code to run at a higher clock frequency, using a larger FPGA, or even application-specific integrated circuit (ASIC) design.

The patch size used in the presented design is a tradeoff between computational complexity and reconstruction accuracy at low percentage of known pixel values. An improvement could be realized by incorporating an adaptive mechanism to increase patch size when only a few pixel intensities are known based on the results of Fig. 7. Although a larger patch size would result in a higher PSNR, it is still questionable whether larger patch sizes would improve reconstruction enough for the reconstruction at low percentages of known pixels to be useful for image processing or object recognition.

In fact, a particular elegant solution that also invokes a multiresolution characteristic is to recover multiple versions of the image or video frame of interest using multiple patch sizes. These various realizations of the scene are then fused together to form the final result. However, due to the tremendous amount of computational complexity as well as memory buffering required, such technique is beyond the scope of this paper. Another possible direction in optimal recovery of visual data from the ATIS is the extension of recovery techniques using sophisticated learned or trained dictionaries [32].

X. Conclusion

We have described a compressive sensing architecture capable of reconstructing data from the ATIS at 304×240 pixel resolution at 28 FPS. The architecture is easily reconfigurable to allow use of different dictionaries or change the number of basis elements used to represent the image. Testing shows that the accuracy achieved by our real-time implementation is comparable to the accuracy achieved by state of the art reconstruction algorithms run offline. We have also outlined possible implementations to run in real time at full HD resolution.

References

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 dB 15 us latency asynchronous temporal contrast vision sensor. Solid-state circuits," *IEEE J.*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[2] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Feb. 2010, pp. 400–401.

[3] M. Mahowald, *An Analog VLSI System for Stereoscopic Vision*. Norwell, MA: Kluwer, 1994.

[4] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," in *Proc. 2010 IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2010, pp. 2426–2429.

[5] U. Mallik, M. Clapp, E. Choi, G. Cauwenberghs, and R. Etienne-Cummings, "Temporal change threshold detection imager," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2005, vol. 1, pp. 362–603.

[6] E. Culurciello, R. Etienne-Cummings, and K. A. Boahen, "A biomorphic digital image sensor. Solid-state circuits," *IEEE J.*, vol. 38, no. 2, pp. 281–294, Feb. 2003.

[7] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan. 2011.

[8] E. Candes and J. Romberg, "Sparsity and incoherence in compressive sampling," *Inverse Problems*, vol. 23, no. 3, pp. 969–969, 2007.

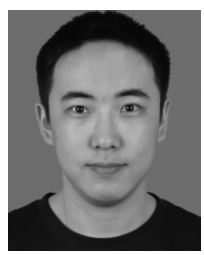
[9] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 83–91, Mar. 2008.
[10] "Rice Single Pixel Camera Project Website," [Online]. Available: http://dsp.rice.edu/cscamera
[11] M. B. Wakin, J. N. Laska, M. F. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. F. Kelly, and R. G. Baraniuk, "Compressive imaging for video representation and coding," in *Proc. Picture Coding Symp. (PCS)*, 2006.
[12] R. Robucci, L. K. Chiu, J. Gray, J. Romberg, P. Hasler, and D. Anderson, "Compressive sensing on a CMOS separable transform image sensor," in *IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2008, pp. 5125–5128.
[13] T. F. Chan and J. Shen, "Image processing and analysis: Variation, PDE, wavelet, and stochastic methods," in *SIAM*, Philadelphia, PA, 2005.
[14] J. Y. Park and M. B. Wakin, "A multiscale framework for compressive sensing of video," in *Picture Coding Symp.*, May 2009, pp. 1–4.
[15] L. W. Kang and C. S. Lu, "Distributed compressive video sensing," in *IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 1169–1172.
[16] J. Zheng and E. L. Jacobs, "Video compressive sensing using spatial domain sparsity," *Opt. Eng.*, vol. 48, no. 8, p. 087006, 2009.
[17] W. L. Chan, K. Charan, D. Takhar, K. F. Kelly, R. G. Baraniuk, and D. M. Mittleman, "A single-pixel terahertz imaging system based on compressed sensing," *Appl. Phys. Lett.*, vol. 93, no. 12, p. 121105, Sep. 2008.
[18] T. T. Do, Y. Chen, D. T. Nguyen, N. Nguyen, L. Gan, and T. D. Tran, "Distributed compressed video sensing," in *Proc. 16th IEEE Int. Conf. Image Process. (ICIP)*, Nov. 2009, pp. 1393–1396.
[19] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid mr imaging," *Magn. Reson. Med.*, vol. 58, no. 6, pp. 1182–1195, 2007.
[20] U. Gamper, P. Boesiger, and S. Kozerke, "Compressed sensing in dynamic MRI," *Magn. Reson. Med.*, vol. 59, no. 2, pp. 365–373, 2008.
[21] H. Jung, K. Sung, K. S. Nayak, E. Y. Kim, and J. C. Ye, "K-t focuss: A general compressed sensing framework for high resolution dynamic MRI," *Magn. Reson. Med.*, vol. 61, no. 1, pp. 103–116, 2009.
[22] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Trans. Inf. Theory*, vol. 52, no. 1, pp. 6–18, 2006.
[23] E. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.
[24] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput*, vol. 20, pp. 33–61, 1998.
[25] S. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.
[26] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.
[27] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230–2249, May 2009.
[28] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *CoRR*, 2009.
[29] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," *ArXiv e-prints*, Sep. 2010.
[30] M. D. Nguyen, D. T. Dao, and T. D. Tran, "Error concealment via 3-mode tensor approximation," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 2081–2084.
[31] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, 2000.
[32] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin, "Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 130–144, Jan. 2012.
[33] D. L. Ruderman and W. Bialek, "Statistics of natural images: Scaling in the woods," *Phys. Rev. Lett.*, vol. 73, pp. 814–817, Aug. 1994.
[34] A. C. Bovik, *The Essential Guide to Image Processing*. New York: Academic, 2009.
[35] International Technology Roadmap for Semiconductors [Online]. Available: http://www.itrs.net

Garrick Orchard received the B.Sc. degree in electrical engineering from the University of Cape Town, Cape Town, South Africa, in 2006 and the M.S.E. and Ph.D. degrees in electrical and computer engineering from Johns Hopkins University, Baltimore, MD, in 2009 and 2012, respectively.

He was a Paul V. Renoff (2007) and a Virginia and Edward M. Wysocki, Sr. (2011) fellow. His research focuses on developing neuromorphic vision sensors and algorithms for real-time sensing on aerial platforms.

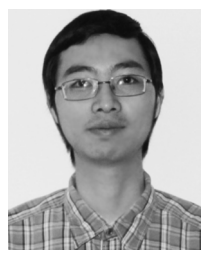
Jie Zhang received the B.Sc. degree in electrical engineering, in 2010, from The Johns Hopkins University, Baltimore, MD, where he is currently pursuing the Ph.D. degree in electrical engineering.

He has been an International Scholar with the Ultra Low Power Circuit group at IMEC, Belgium, from October 2011 to July 2012. The focus of his research is compressive sensing, analog and mixed signal circuits with low power biomedical applications.



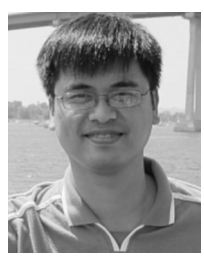
Yuanming Suo received the B.S. degree in electronic information science and technology from Sun Yat-sen University, Guangzhou, China, in 2004, and the M.S.E. degree in electrical and computer engineering from University of Alabama, Huntsville. Currently, he is a Ph.D. degree student at Johns Hopkins University, Baltimore, MD, in the Digital Signal Processing Lab, where his research focuses on developing algorithms for video concealment and recovering missing information in large matrices.

From 2004 to 2007, he worked at China Netcom as a Computer Network Engineer.



Minh Dao received the B.Sc. degree in electrical engineering from Hanoi University of Technology, Hanoi, Vietnam, in 2007, and the M.Sc. degree in information and communication technologies from both Polytechnic University of Turin, Turin, Italy, and Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2009. He is currently pursuing the Ph.D. degree in electrical and computer engineering at The Johns Hopkins University, Baltimore, MD.

His research interests include sparse signal representation, sparse recovery, and compressed sensing with the current focus on low-rank matrix completion, and robust principal component analysis.



Dzung T. Nguyen received the B.E. degree in electronics and telecommunications from Hanoi University of Technology, Hanoi, Vietnam, in 2002, and the M.S.E degree, in 2008, from the Johns Hopkins University, Baltimore, MD, where he is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering.

His research interests are on image/video/high dimensional data processing and unsupervised target detection.

Sang (Peter) Chin received the B.S. degree (*Phi Beta Kappa*) in electrical engineering, computer science, and mathematics from Duke University, Durham, NC, and the Ph.D. degree in mathematics from the Massachusetts Institute of Technology, Cambridge.

He is currently the branch Chief Scientist of the Cyber Technology Branch at Johns Hopkins University (JHU) Applied Physics Laboratory. His main research activities lie in the areas of compressive sensing, data fusion, game theory, multiple hypothesis tracking, quantum-game inspired cyber-security, sensor resource management, and cognitive radio. He is currently PI for a four-year ONR grant (Geometric Multi-Resolution Analysis) where he is applying geometric sparse recovery techniques to high dimensional data with low intrinsic dimension. He is a Merle A. Tuve fellow as an Adjunct Assistant Research Professor in the Electrical and Computer Engineering Department in the Whiting School of Engineering at JHU and is also an adjunct Associate Professor of mathematics at University of Maryland Baltimore County.

Christoph Posch received the M.Sc. and Ph.D. degrees in electrical and electronics engineering and experimental physics from Vienna University of Technology, Vienna, Austria, in 1995 and 1999, respectively.

From 1996 to 1999, he worked on analog CMOS and BiCMOS IC design for particle detector readout and control at CERN, the European Laboratory for Particle Physics in Geneva, Switzerland. From 1999 onwards he was with Boston University, Boston, MA, engaging in applied research and analog/mixed-signal integrated circuit design for high-energy physics instrumentation. In 2004 he joined the newly founded Smart Sensors Group at AIT Austrian Institute of Technology (formerly Austrian Research Centers ARC) in Vienna, Austria, where he was promoted to Principal Scientist in 2007. Since 2012, he is with the Institut de la Vision at Université Pierre et Marie Curie in Paris, France. His current research interests include neuromorphic analog VLSI, CMOS image and vision sensors, and biology-inspired signal processing. He has authored more than 70 scientific publications and holds several patents in the area of vision and image sensing.

Dr. Posch has been recipient and co-recipient of several scientific awards including the Jan van Vessem Award for Outstanding European Paper at the IEEE International Solid-State Circuits Conference (ISSCC) in 2006, the Best Paper Award at ICECS 2007, and Best Live Demonstration Awards at ISCAS 2010 and BioCAS 2011. He is a member of the Sensory Systems and the Neural Systems and Applications Technical Committees of the IEEE Circuits and Systems Society.

Trac D. Tran (S'94–M'98–SM'08) received the B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1993 and 1994, respectively, and the Ph.D. degree in electrical engineering from the University of Wisconsin, Madison, in 1998.

In July of 1998, he joined the Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD, where he currently holds the rank of Associate Professor. His research interests are in the field of digital signal processing, particularly in sparse representation, sparse recovery, sampling, multirate systems, filter banks, transforms, wavelets, and their applications in signal analysis, compression, processing, and communications. His pioneering research on integer-coefficient transforms and pre/postfiltering operators has been adopted as critical components of Microsoft Windows Media Video 9 and JPEG XRC the latest international still-image compression standard ISO/IEC 29199–2. In the summer of 2002, he was an ASEE/ONR Summer Faculty Research Fellow with the Naval Air Warfare Center C Weapons Division at China Lake, CA. He is currently a Regular Consultant for the U.S. Army Research Laboratory in Adelphi, MD.

Dr. Tran was the co-Director (with Prof. J. L. Prince) of the 33rd Annual Conference on Information Sciences and Systems (CISS'99), Baltimore, in March 1999. He has served as Associate Editor of the IEEE TRANSACTION ON SIGNAL PROCESSING, as well as IEEE TRANSACTIONS ON IMAGE PROCESSING. He was a former member of the IEEE Technical Committee on Signal Processing Theory and Methods and is a current member of the IEEE Image Video and Multidimensional Signal Processing Technical Committee. He received the NSF CAREER award in 2001, the William H. Huggins Excellence in Teaching Award from The Johns Hopkins University in 2007, and the Capers and Marion McDonald Award for Excellence in Mentoring and Advising in 2009.

Ralph Etienne-Cummings (F') received the B.S. degree in physics, from Lincoln University, Pennsylvania, in 1988, and the M.S.E.E. and Ph.D. degrees in electrical engineering from the University of Pennsylvania, in December 1991 and 1994, respectively.

Currently, he is a Professor of electrical and computer engineering, and computer science at Johns Hopkins University (JHU), Baltimore, MD. He is the former Director of Computer Engineering at JHU and the Institute of Neuromorphic Engineering (currently administered by University of Maryland, College Park). He is also the Associate Director for Education and Outreach of the National Science Foundation (NSF) sponsored Engineering Research Centers on Computer Integrated Surgical Systems and Technology at JHU. His research interest includes mixed signal VLSI systems, computational sensors, computer vision, neuromorphic engineering, smart structures, mobile robotics, legged locomotion and neuroprosthetic devices.

Dr. Etienne-Cummings has served as Chairman of the IEEE Circuits and Systems (CAS) Technical Committee on Sensory Systems and on Neural Systems and Application, and was re-elected as a member of CAS Board of Governors from 2007–2009. He was also the General Chair of the IEEE BioCAS 2008 Conference. He was also a member of Imagers, MEMS, Medical and Displays Technical Committee of the ISSCC Conference from 1999–2006. He is the recipient of the NSF's Career and Office of Naval Research Young Investigator Program Awards. In 2006, he was named a Visiting African Fellow and a Fulbright Fellowship Grantee for his sabbatical at University of Cape Town, South Africa. He was invited to be a lecturer at the National Academies of Science Kavli Frontiers Program, held in November 2007. He has also won publication awards, including the 2003 Best Paper Award of the EURASIP Journal of Applied Signal Processing and "Best Ph.D. in a Nutshell" at the IEEE BioCAS 2008 Conference, and has been recognized for his activities in promoting the participation of women and minorities in science, technology, engineering, and mathematics.