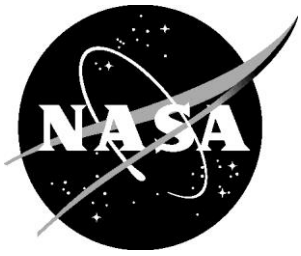
NASA/CR–2015-219000



COxSwAIN: Compressive Sensing for Advanced Imaging and Navigation

Richard Kurwitz, Marina Pulley, Nathan LaFerney, and Carlos Munoz
Texas A&M University, College Station, Texas

December 2015

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

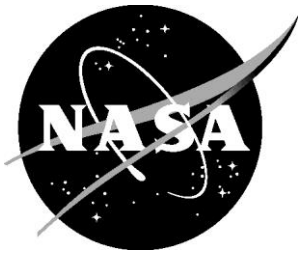
- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/CR–2015-219000



COxSwAIN: Compressive Sensing for Advanced Imaging and Navigation

Richard Kurwitz, Marina Pulley, Nathan LaFerney, and Carlos Munoz
Texas A&M University, College Station, Texas

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Cooperative Agreement NNX13AR50A

December 2015

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Abstract

The COxSwAIN project focuses on building an image and video compression scheme that can be implemented in a small or low-power satellite. To do this, we used Compressive Sensing, where the compression is performed by matrix multiplications on the satellite and reconstructed on the ground. Our paper explains our methodology and demonstrates the results of the scheme, being able to achieve high quality image compression that is robust to noise and corruption.

1 Introduction

1.1 Objectives

Goal #1: Develop a compression and reconstruction scheme for images. *This goal has been met.* A set of MATLAB codes have been developed to compress and reconstruct images. Separate codes have been provided for the compression and reconstruction of images.

Goal #2: Devlop a compression and reconstruction scheme for video. *This goal has been partially met.* Though a set of codes has been developed to perform video compression and reconstruction, compression in real-time was never explored. Real-time reconstruction of video after compression was never tested but would most likely not work since the reconstruction of images typically takes a few minutes.

Goal #3: Explore and adjust quality of the scheme. *This goal has been met.* Various factors have been studied in the image compression and reconstruction, including different settings that can be employed during the compression or reconstruction phases of the scheme. These include the effects of varying amounts of compression, data types of the compressed images and video, different settings used during reconstruction and the effects of noise and corruption on the reconstruction.

1.2 Overview

The COxSwAIN project is concerned with developing an image or video compression scheme that could be implemented on a small or low-power satellite. To carry this out, Compressive Sensing is used since the compression can be performed by a set of matrix multiplications. Though the reconstruction of the compressed image or video can be computationally expensive, reconstruction can be performed on the ground.

The COxSwAIN project originally began as a hardware based approach, where the compression would be performed by multiplexing the

image. As time went on, the project shifted to a software based approach with code being written in MATLAB. This paper outlines and gives results demonstrating our approach. The paper is organized as follows: a short background on Compressive Sensing and an explanation of the reconstruction process is provided in Section 3. An overview of the codes written to process image and video is provided in Section 4. Lastly, results are provided in Section 5.

2 Background

2.1 A Short Introduction to Compressive Sensing and the Compression of Signals

Compressive Sensing is a relatively young area of signal processing that deals with the the compression of linearly modeled signals; for example, images. Compressive Sensing performs compression by taking non-adaptive linear measurements of a signal. One of the goals in Compressive Sensing, and in our own research, is to find the minimum number of measurements needed to perform a reconstruction that is (near) perfect.

Suppose that $\mathbf{x} \in \mathbb{R}^n$ is our signal of interest. We can compress $\mathbf{x}$ by multiplying $\mathbf{x}$ by an $m \times n$ matrix Φ, where $m \ll n$. Letting

$$\mathbf{y} = \Phi\mathbf{x}, \tag{1}$$

$\mathbf{y}$ represents our compressed signal. Basic Linear Algebra tells us that this system has infinitely many solutions. However, when Φ and $\mathbf{x}$ meet certain conditions, then the original signal can be recovered.

In order to perform Compressive Sensing, we must choose a Φ matrix that satisfies the *Restricted Isometry Property* (RIP). A matrix Φ is said to satisfy the RIP of order $k \in \mathbb{N}$ if there exists $\delta_k \in (0, 1)$ such that

$$(1 - \delta_k)\|\mathbf{x}\|_2 \leq \|\Phi\mathbf{x}\|_2 \leq (1 + \delta_k)\|\mathbf{x}\|_2 \tag{2}$$

for any $\mathbf{x} \in \mathbb{R}^n$ such that $\|\mathbf{x}\|_0 \leq k$, where $\|\cdot\|_0$ denotes the sparsity, or the number of non-zero entries in a vector.[1] Intuitively, the RIP states that Φ approximately preserves the distance between sparse vectors, the same way that if Φ was an invertible matrix.

The Φ matrices used in our particular application consists of entries drawn from a Normal distribution and then orthonormalized. These are very common CS matrices and often appear in literature. It can be shown that Φ satisfies the RIP and

$$m \leq Ck \log\left(\frac{n}{k}\right), \tag{3}$$

where C is an arbitrary constant, with high probability. Such a formula is useful for showing that Φ makes a good compressive sensing matrix,

[1] It should be noted that $\|\cdot\|_0$ does not actually satisfy the properties of a norm, and therefore should not be considered as one.

using it to determine the value of m is not as clear. Therefore we have relied heavily on empirical results to demonstrate the effectiveness of our approach when used for image compression.

For a more detailed explanation and proofs of the previous statements, we direct the reader to [3]

In our own approach, instead of directly compressing an image, we split the image into distinct square blocks and each block is compressed individually using the same Φ matrix. This approach is known as Block-based Compressive Sensing (BCS). Computationally, reconstruction of our image in this compressed form becomes easier. In the following subsection, we will outline how we reconstruct our image when we employ BCS.

2.2 Image Reconstruction in a Compressive Sensing Framework

In practice, it can be very difficult to find the unique sparse solution to (1). Instead, we relax the problem and instead try to solve the convex-optimization problem

$$\min_{\mathbf{x}} \|\mathbf{y} - \Phi\mathbf{x}\|_2 + \|\mathbf{x}\|_1. \tag{4}$$

This formulation allows us to find a sparse solution to that is "close" to (1) instead of searching for it directly, making the computation feasible[2].

The algorithm we employ for the reconstruction of compressed images is the Multiple Hypothesis Block-based Compressive Sensing with Smooth Projected Landweber (MH-BCS-SPL) [4] [5] [6]. MH-BCS-SPL works by employing a Smooth Projected Landweber iteration to generate an intial reconstruction, then Multi-Hypothesis (MH) predictions are employed to improve the quality of the reconstruction. Thus, when describing MH-BCS-SPL, we can break it up into two sections: The SPL step and the MH step. Then a summary of the algorithm will be provided.

2.2.1 Smooth Projected Landweber

When performing an SPL step, we first compute an initial guess to perform the reconstruction. We used $x^{(0)} = \Phi^T\mathbf{y}$ for an initial guess, though in theory, another image could be used as an input. A Wiener filter is employed to smooth the image and reduce the artifacts from blocking. Then we compute

$$\hat{\mathbf{x}}^{(i)} = \mathbf{x}^{(i-1)} + \Phi^T(\mathbf{y} - \Phi\mathbf{x}^{(i-1)}). \tag{5}$$

The vector $\hat{\mathbf{x}}^{(i)}$ is then multiplied by a change of basis matrix Ψ and Bivariate Shrinkage [7] is applied toΨ $\hat{\mathbf{x}}^{(i)}$ to promote a sparse solution

[2]Searching for a sparse solution is NP-Hard.

in the Ψ basis. ApplyΨ^{-1} to obtain $\tilde{\mathbf{x}}^{(i)}$ and then compute

$$\mathbf{x}^{(i)} = \tilde{\mathbf{x}}^{(i)} + \Phi^T(\mathbf{y} - \Phi\tilde{\mathbf{x}}^{(i)}). \tag{6}$$

We perform this iteration for k_{SPL} steps or until Residual Means Square of $x^{(i+1)}$ and $x^{(i)}$ is below the threshold ϵ, where in practice $k_{SPL} = 200$ and $\epsilon = 0.0001$. The λ value is used to control the Bivariate Shrinking and will be discussed more in Section 4.1.

The algorithm is summarized as follows:

Algorithm 1 SPL

Require: $\mathbf{y}, \Phi, \Psi, \lambda, \epsilon, k_{SPL}$

$\mathbf{x}^{(0)} = \Phi^T\mathbf{y}$
for $i = 1$ to k_{SPL} **do**
  $\mathbf{x}_W = Wiener(\mathbf{x}^{(i-1)})$
  $\hat{\mathbf{x}} = \mathbf{x}_W + \Phi^T(\mathbf{y} - \Phi\mathbf{x}_W)$
  $\mathbf{z} = \Psi\hat{\mathbf{x}}$
  $\bar{\mathbf{z}} = BivShr(\mathbf{z})$
  $\tilde{\mathbf{x}} = \Psi^{-1}\bar{\mathbf{z}}$
  $\bar{\mathbf{x}} = \tilde{\mathbf{x}} + \Phi^T(\mathbf{y} - \Phi\tilde{\mathbf{x}})$.
  if $RMS(\bar{\mathbf{x}}, \mathbf{x}^{(i-1)}) \leq \epsilon$ **then**
    Break
  end if
  $\mathbf{x}^{(i)} = \bar{\mathbf{x}}$
end for
return $\bar{\mathbf{x}}$

It is important to note here that we do not use the entire compressed image $\mathbf{y}$ but instead withhold a set to perform Cross-Validation techniques [8]. The holdout set that we use in our algorithm is the entries or rows of $\mathbf{y}$ corresponding to the last three rows of Φ when multiplying $\mathbf{x}$. These entries will be used as a comparison when performing Multi-Hypothesis Predictions to measure and improve the accuracy of the reconstruction.

2.2.2 Multi-Hypothesis Predictions

The idea with performing a MH Prediction in our algorithm is we cull information from within a block to improve the accuracy of the reconstruction.

For each block of the initial reconstruction, provided by the $BCS - SPL(\cdot)$ function, we will symmetrically extend each block of $\bar{\mathbf{x}}$ by w pixels to create a search window that is of size $(B + 2w) \times (B + 2w)$ pixels, denoted by $\hat{\bar{\mathbf{x}}}_i$ Let $\hat{\bar{\mathbf{x}}}_{i,j}$ a subblocks of $\hat{\bar{\mathbf{x}}}_i$, $j = 1 \ldots N$, N being the number of subblocks.

For every subblock $\hat{\bar{\mathbf{x}}}_{i,j}$, construct a column vector $\mathbf{h}_j$ with B^2 entries. The entries of $\mathbf{h}_j$ consists of the pixel values from $\bar{\mathbf{x}}_{i,j}$, the remaining

ones zero, and each entry of $\mathbf{h}_j$ corresponds to a pixel of $\bar{\mathbf{x}}_i$. Let $\mathbf{H}_i = [\mathbf{h}_1, \dots \mathbf{h}_N]$. We will use $\mathbf{H}_i$ to compute

$$\hat{\mathbf{w}}_i = \min_{\mathbf{w}} \|\mathbf{y}_i - \Phi\mathbf{H}_i\mathbf{w}\|_2^2 + \|\Gamma\mathbf{w}\|_2^2, \tag{7}$$

which can be found by computing

$$\hat{\mathbf{w}}_i = \left((\Phi\mathbf{H}_i)^T(\Phi\mathbf{H}_i) + \Gamma^T\Gamma\right)^{-1}(\Phi\mathbf{H}_i)^T\mathbf{y}_i \tag{8}$$

The matrix Γ is a diagonal matrix, with the nonzero entries found by $\|\mathbf{y}_i - \Phi\mathbf{H}_j\|_2^2$ and is an example of a Tikhonov Matrix. The intuition behind this approach is that we do not want to assign as much weight to subblocks that differ more significantly than others.

The algorithm is summarized as follows:

Algorithm 2 MH

Require: $\mathbf{y}, \Phi, \bar{\mathbf{x}}, B, b, w$

1: **for** $i = 1$ to M **do**
2: **for** $j = 1$ to N **do**
3: Construct H_j
4: **end for**
5: ConstructΓ
6: $\hat{\mathbf{w}} = \left((\Phi\mathbf{H}_i)^T(\Phi\mathbf{H}_i) + \Gamma^T\Gamma\right)^{-1}(\Phi\mathbf{H}_i)^T\mathbf{y}_i$
7: **end for**
8: **return** $\hat{\mathbf{w}}$.

As stated in Section 3.2.1, we have a holdout set that is used to measure the performance. We will denote the sets we used for the reconstruction and the holdout will be denoted byΦ_R andΦ_H respectively. Therefore $\Phi = [\Phi_R; \Phi_H]$. The algorithm will be summarized, and then explained for ease of understanding.

Essentially, the algorithm begins by computing an initial reconstruction with the SPL function. Then we perform Multi-Hypothesis Predictions and residual reconstructions using SPL are performed to further improve the quality of the reconstruction. When performing the MH predictions, we set $b = \frac{B}{2}$ and $w = 5$. Structural Similarity (SSIM) [9] is used to measure the similarity of the current and previous reconstructed images, though in theory a measure such as RMS could be used instead. A residual is taken of the reconstructed image compressed byΦ_H and $\mathbf{y}_H$. The intuition behind this is that if the reconstructed image using Φ_R is close to the original image, then the $R^{(i)}$ value should be relatively small. If $R^{(i)}$ (as a function of i) begins to decrease and the reconstructed image begins to converge to its solution, then we can increase the size of the search window and subblock size to the actual block size, and more iterations are performed. Increasing the subblock size is performed so that the solution does not introduce any block artifacts, thus the search window must be increased in proportion. If $R^{(i)}$ begins to increase when

Algorithm 3 MH-BCS-SPL

Require: $\mathbf{y}, \Phi, B, b, \Psi, \lambda, \epsilon, \tau, k_{SPL}, k_{MH}$
  $\mathbf{x}^{(0)} = SPL(\mathbf{y}_R, \Phi_R, \Psi, \lambda, \epsilon, k_{SPL})$
  for $i = 1$ to k_{MH} **do**
    $\tilde{\mathbf{x}} = MH(\mathbf{y}_R, \Phi_R, \mathbf{x}^{(i-1)}, B, b)$
    $\hat{\mathbf{x}} = \tilde{\mathbf{x}} + SPL(\mathbf{y}_R - \Phi_R\tilde{\mathbf{x}}, \Phi_R, \Psi, \lambda, \epsilon, k_{SPL})$
    $s^{(i)} = SSIM(\hat{\mathbf{x}}, \mathbf{x}^{(i-1)})$
    $R^{(i)} = \|\mathbf{y}_H - \Phi_H\hat{\mathbf{x}}\|_2$
    if $b < B$ **then**
      if $R^{(i)} < R^{(i-1)}$ and $|s^{(i)} - s^{(i-1)}| < \tau$ **then**
        $b \leftarrow B$
        $w \leftarrow 2 \times w$
      end if
    end if
    if $R^{(i)} > R^{(i-1)}$ and $B = b$ **then**
      Break
    end if
    $\mathbf{x}^{(i)} = \hat{\mathbf{x}}$
  end for
  return $\hat{\mathbf{x}}$

$B = b$, then this is an indicator that the algorithm may be diverging, so the algorithm is terminated.

3 Image and Video Processing

3.1 Image Code

The image code begins by inputting an image. The first step is to divide the image into blocks. So that our code can accept images of multiple sizes, we compute the block size by taking the greatest common denominator of the width and height of the image in pixels.

To perform the compression, a Φ matrix must be generated. First, the size of Φ must be chosen. If Φ is an $m \times n$ matrix, n is chosen based on the blocksize. So if the size of a block is $B \times B$, then $n = B^2$. To determine m, we choose a number s such that $0 < s < 1$ and set $m = sn$.[3] We refer to the value s as the *subrate*. Since $s = \frac{m}{n}$, the subrate can be thought of the number of measurements taken relative to the size of the image, and thus can act as a measure of compression. We generate Φ by creating an $n \times n$ matrix consisting of entries drawn from a Normal Distribution, orthonormalizing it and choosing m rows of the matrix. We then multiply our image $\mathbf{x}$ by Φ to compress our image. The compressed image $\mathbf{y}$ may then be truncated from entries consisting of double precision floats to single-precision floats or 16-bit integers. In the

[3] Obviously we will round the value of sn and will be greater than zero.

results section, we analyze the effects of using different these different data types on the quality of reconstruction.

After the image is compressed, we can invoke the MH-BCS-SPL algorithm to reconstruct our image (see Section 3.2 for the overview of this). As stated previously, we perform the reconstruction using a Dual Discrete Wavelet Transform (DDWT). However, the user can invoke additional arguments when running the function to use different transforms. These include the Discrete Wavelet Transform, Discrete Cosine Transform, Hadamard Transform, and option to not use a basis at all (so the Bivariate Shrinkage step is skipped). The level of Bivariate Shrinkage can be controlled as well by specifying the λ value that is used. If the user does not specify, then a default value of $\lambda = 6$ is used. Since the compression and reconstruction are independent of each other, the user could in theory use all transforms and settings until one is found that gives optimal results.

In the case that we have a color image that is an RGB or CMYK, we simply take each color channel and treat it separately as an individual image. After reconstructing each color channel, the channels are simply combined back to obtain our entire reconstructed image.

3.2 Video Code

When processing video, the video can be split into a sequence of frames. Thus, we simply perform the compression on each frame individually, simply applying our code to process images repeatedly. Each compressed frame is not saved however; instead we take save a set of frames as key frames. For each key frame, the following frames, until another key frame is reached, are saved as residuals between it and the sets key frame. For example, if we save every $10th$ frame as a key frame in a set of frames $\{\mathbf{y}_1, \dots \mathbf{y}_t\}$, then for every i, $i = 1, \dots t$, if $i \not\equiv 0 (\text{mod} 10)$, then $\mathbf{y}_i$ is saved as $\mathbf{y}_i^* = \mathbf{y}_{i-i^*} - \mathbf{y}_i$, where $i^* = i (\text{mod } 10)$. In our own algorithm, we simply save the first frame as the only key frame, and the remainder we save as residuals.

There are two advantages to this approach. The first is that resulting set of compressed frames will be much more compressible since many of the coefficients will be zero or close to zero. The other advantage is speeding up the video reconstruction since we can simply reconstruct the residuals and add it back to the reconstructed key frame. Since many of the entries of would be zero, the MH-BCS-SPL algorithm can process them quickly.

4 Results

4.1 Results: Basic

In this section, basic results will be reviewed and discussed. A variety of images were tested, including high definition images, Infrared images, as well as gray scale images. When the image is compressed, we can save the compressed image using different precisions. Our results will show the results for the compressed image in double precision, single precision and 16-bit integers. The images displayed in the section will be the original images and the reconstructed image where the compressed image is 16-bit integers. All images were compressed using a .1 subrate and reconstructed with a DDWT basis and $\lambda = 6$.

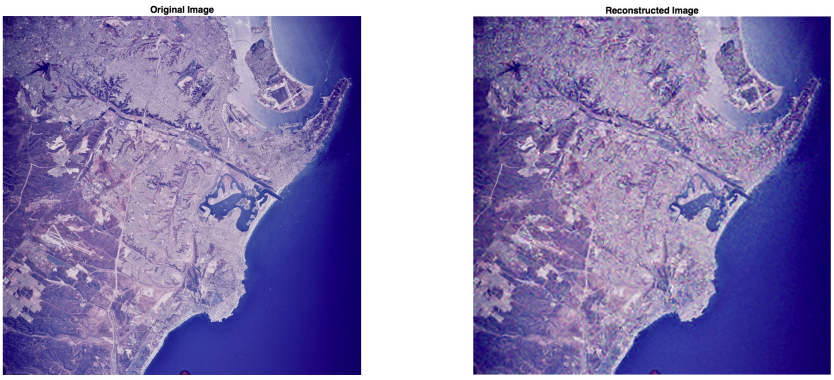


Figure 1. The *Land* image shown in Figure 1 is a color image of size 1024×1024.

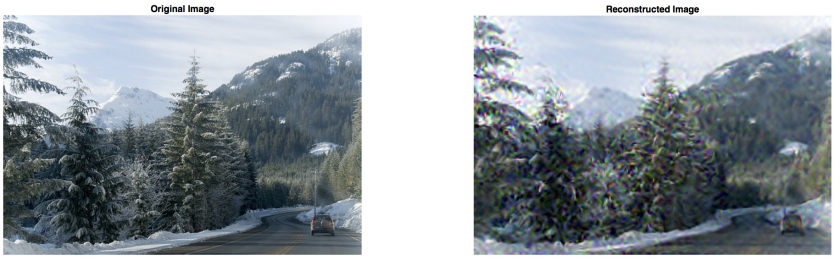


Figure 2. The *Forest* image shown in Figure 2 is a color image of size 400×600.

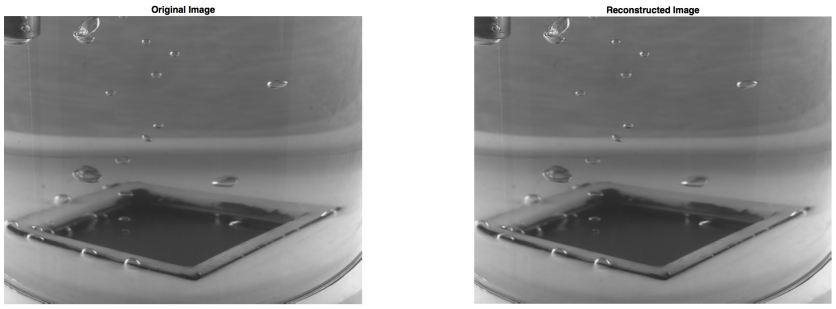


Figure 3. The *Bubbles* image shown in Figure 3 is a color image of size 1024×1280.

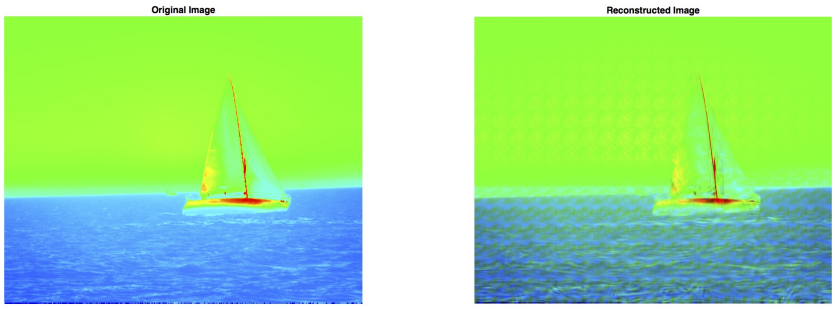


Figure 4. The *Sail* image shown in Figure 4 is an Infrared image of size 512×640.

Figure 5. The *Peppers* image shown is a greyscale image of size 512×512.

Measure	Land	Forest	Bubbles	Sail	Peppers
Double CR	.7969	.8000	.7969	.7969	.7969
Double PSNR	21.7042	21.0017	43.9455	16.2068	30.2580
Single CR	.3984	.4000	.3984	.3984	.3984
Single PSNR	21.7042	21.0017	43.9455	16.2068	30.2580
Int16 CR	.1992	.2000	.1992	.1992	.1992
Int16 PSNR	21.7038	21.0083	43.8832	16.2847	30.2559

Typically, there are features present in images that lend itself to Compressive Sensing. Images with fewer areas of high-contrast will have better reconstructions. This however is to be expected since the reconstruction is sparse within a Wavelet domain, thus edges are usually smoothed. The *Land* image is a good example of this, but this effect is especially present in the *Sail* picture, especially on the waves in the image. On the other hand, *Bubbles* is a smooth image, thus the reconstruction has especially good results.

One of the features that lends itself well to reconstruction is larger images will typically reconstruct better than smaller ones. This can already be seen by some of the results here, but in our own tests, larger images will always have better PSNR scores as well as less smoothing along the edges. Very small images (e.g. 64×64) will typically collapse under noise or have strange artifacts (an example of such artifacts will be seen when different bases are used to reconstruct images).

Unfortunately, our algorithm does not beat JPEG in terms of compression. To achieve image compressions that are comparable to JPEG requires to use less compression, thus JPEG easily outshines our own method. However, the following subsections will outline the advantage of our method over JPEG as well as results against MPEG video compression which our method is superior to in terms of compression.

4.2 Results: Robustness Under Noise and Corruption

In this section, we will examine the reconstruction process in the presence of noise. There are two cases of noise and corruption that we are concerned with, noise and corruption before compression and after. Again, the compressed images were saved as 16-bit integers and reconstructed using a DDWT basis and $\lambda = 6$.

4.2.1 Gaussian Noise

Gaussian noise in the original or compressed image is created by adding a number drawn from a Normal Distribution to the matrix.

Figure 6. Gaussian Noise Before Compression.

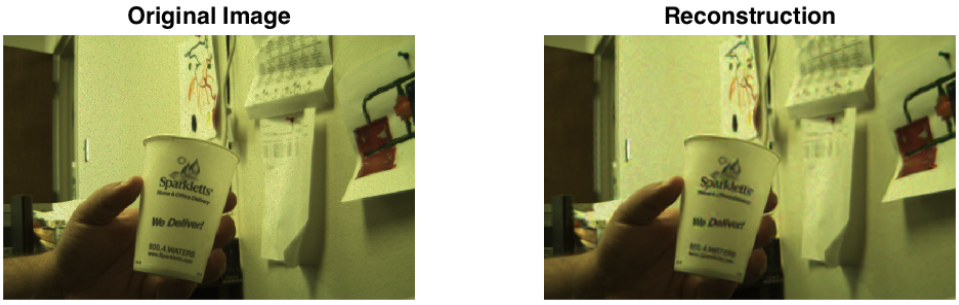


Figure 7. Gaussian Noise After Compression.

4.2.2 Corruption

This subsection studies the effects of corruption on the reconstruction process of images. The same settings during the reconstruction used in the previous section are used in this section as well.

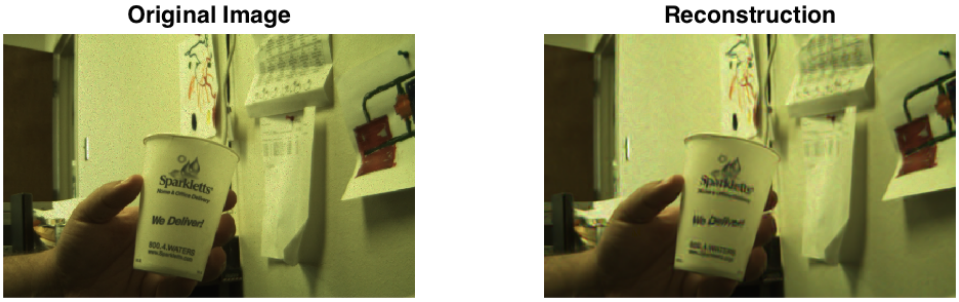


Figure 8. The corruption present in this reconstruction is when a random entry in the compressed image matrix is set to zero.

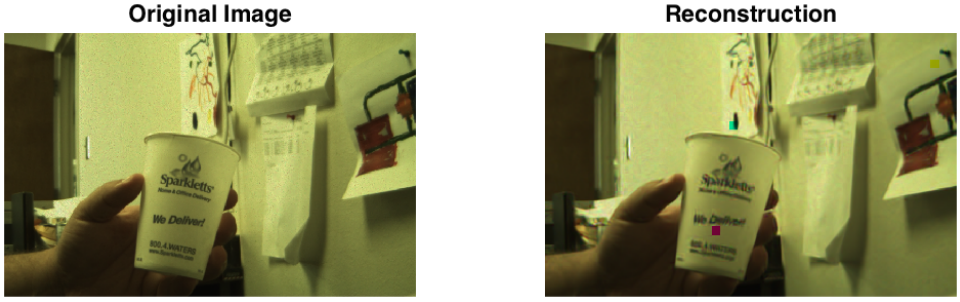


Figure 9. The corruption present in this reconstruction is that a block of the compressed image has been set to all zeroes.

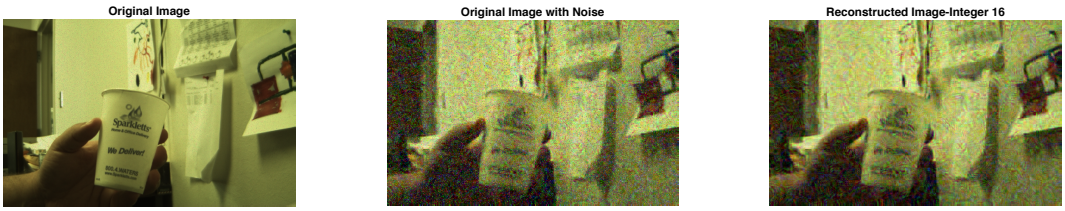


Figure 10. The corruption present in this image is that 5% of the pixels in the image have been randomly chosen and set to zero.

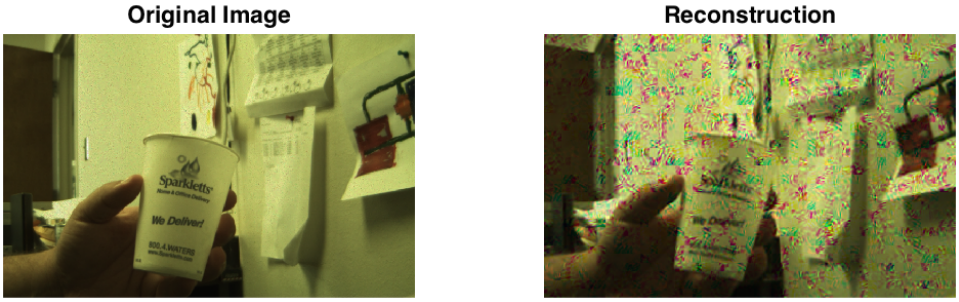


Figure 11. The corruption present in this image is that 5% of the entries of the compressed image have been randomly chosen and set to zero.

4.3 Application of Different Bases

Additional experiments were performed with the reconstruction employing different bases to perform the thresholding in. The basis used will promote a sparse solution in whatever basis we perform the thresholding in (which is necessary to perform the reconstruction). We employed five different options: Dual-Discrete Wavelet Transform (DDWT), Discrete Wavelet Transform (DWT), Discrete Cosine Transform, Hadamard Transform and no thresholding at all, which we will refer to as the Standard Basis option. The first three have been used in Image Processing for many years and images are often sparse or compressible when using one of these transforms. The other two were a result of our curiosity; the Hadamard Transform is just a series of high-pass filters and thus very different from the wavelet transforms while the lack of thresholding was used to test the significance of the thresholding in the reconstruction process. Another interest was to see if any improvement could be made on the sharp edges of a reconstructed image, which are often distorted or smoothed under the reconstruction process. A Canny Filter was used to display the edges of the image. Below the figures is a table with the PSNR scores for each.

The PSNR scores of the Canny Filter should be taken with a grain of salt however. The PSNR score can vary wildly by deciding how to represent the white pixels. In our case, we decided, to set them to a

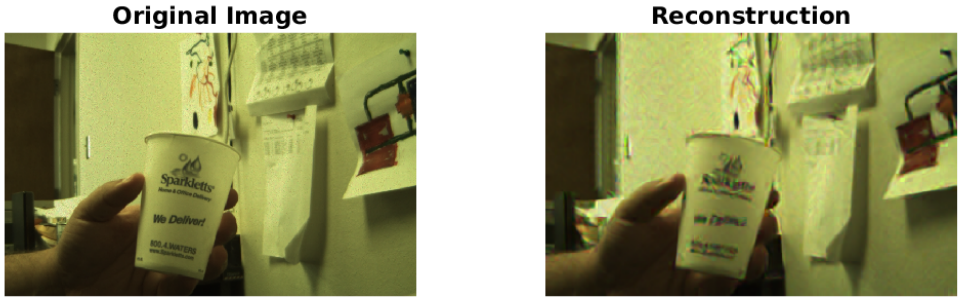


Figure 12. Dual-Discrete Wavelet Transform

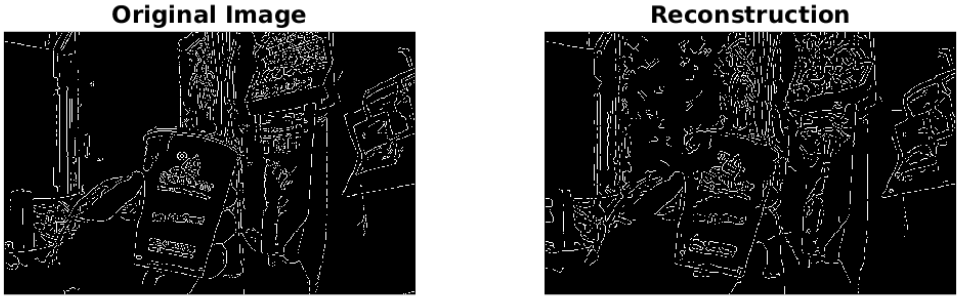


Figure 13. Dual-Discrete Wavelet Transform with the Canny Filter

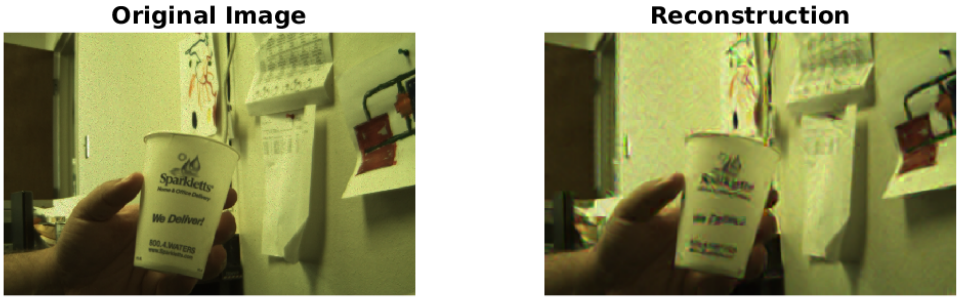


Figure 14. Discrete Wavelet Transform

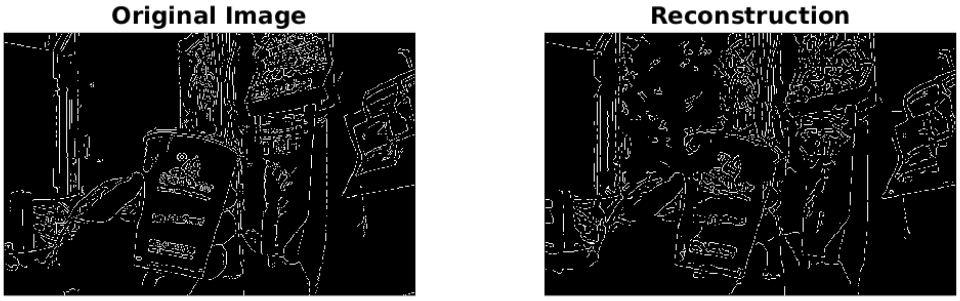


Figure 15. Discrete Wavelet Transform with the Canny Filter

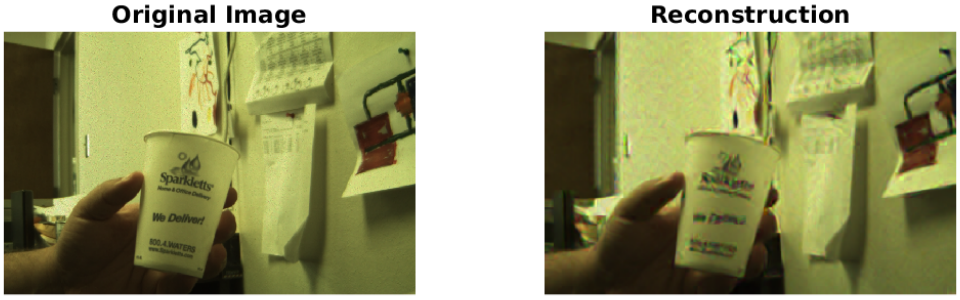


Figure 16. Discrete Cosine Transform

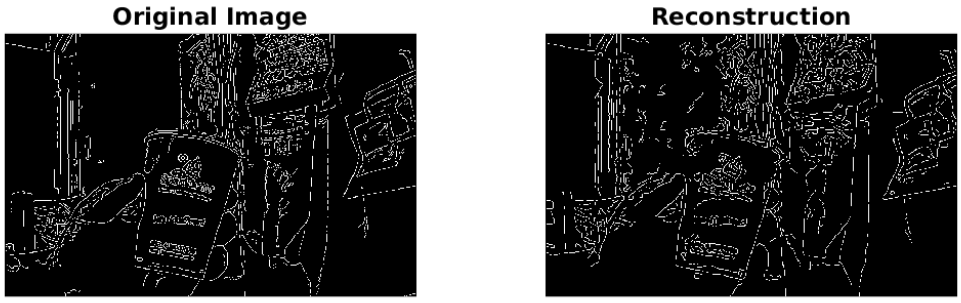


Figure 17. Discrete Cosine Transform with the Canny Filter

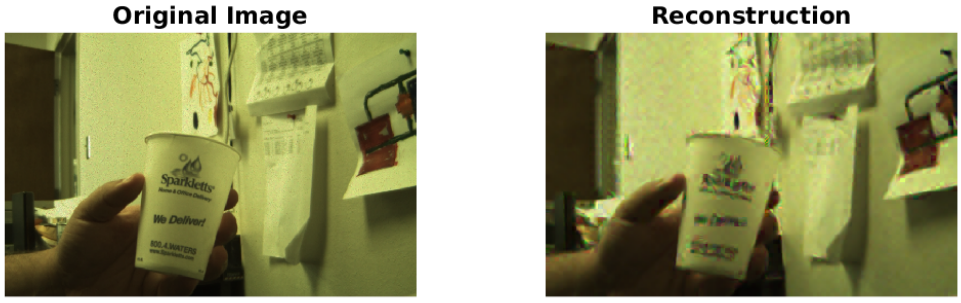


Figure 18. Hadamard Transform

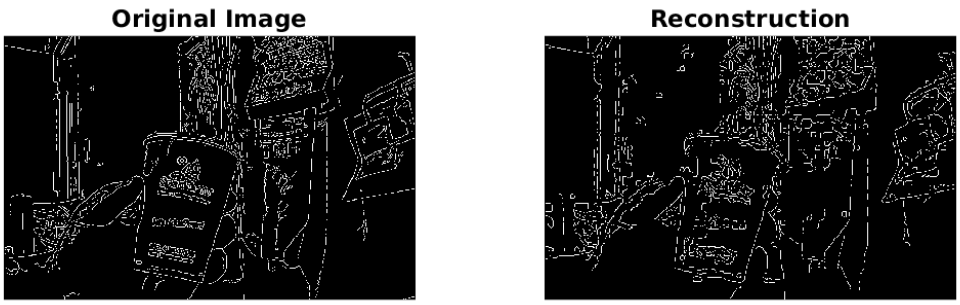


Figure 19. Hadamard Transform with the Canny Filter

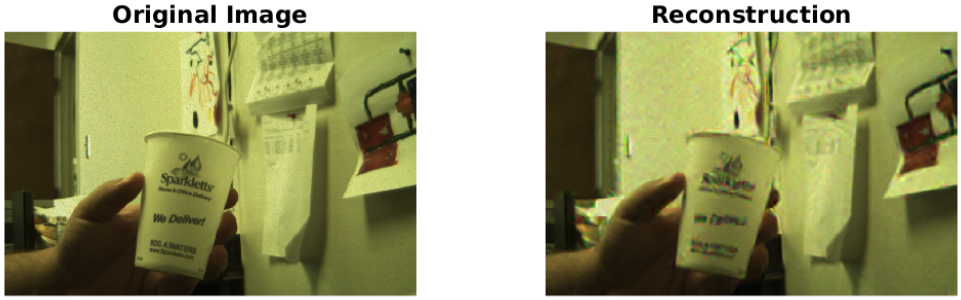


Figure 20. Standard Basis

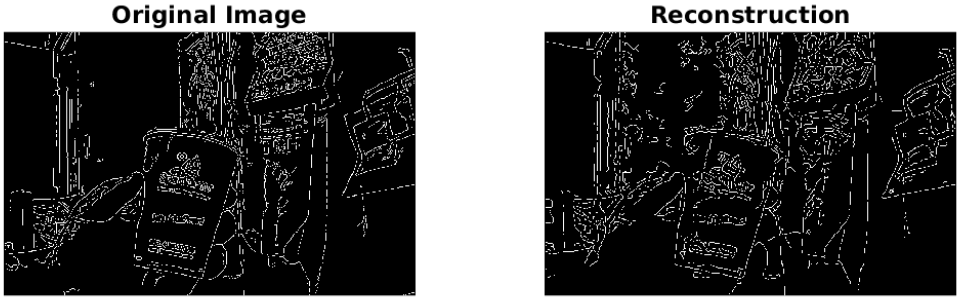


Figure 21. Standard Basis with the Canny Filter

Measure	DDWT	DWT	DCT	Had	SB
Reconstructed	29.6334	29.6435	29.7554	28.7966	29.5982
Canny	11.1114	11.0633	11.2189	10.7481	11.1164

Greyscale value of 255. Measuring the difference in edges is a difficult problem in Image Processing and we decided to take the simplest solution and the PSNR scores to compare the images against each other.

4.4 Video Results

It is very difficult to quantify the quality of a reconstructed video. One could do a frame-by-frame comparison between the original and reconstructed videos, similar to what was done with images. However, a spreadsheet of PSNR values does little to display the quality of a reconstructed video. It is worth mentioning however that by reconstructing the residuals of the compressed frame instead of the entire compressed frame speeds up the process significantly and has little to no effect on the quality of the reconstruction.

The amount of compression performed is one thing that is easy to quantify and can be easily compared against is the size of the compressed video using MPEG2[4] and the size using our algorithm. A small AVI video file consisting of thirty-seven frames and a file size of 298.5 kilobytes was chosen and compressed using the two competing methods. The MPEG2 version of the video had a file size of 79.9 kilobytes and a compression ratio of about 27%. Our algorithm compressed the video to about 52.3 kilobytes when saved as a *.mat* file and achived a compression ratio of about 18%.

5 Conclusion

Matalb was used to create two separate codes to deal with image and video compression and reconstruction. The layout of the image and video code were discussed, as well as the results. The basic results show a variety of images. This was done to show how the code can handle and process everything from small to large images, as well as color and grayscale. In the reconstructions, different features can be seen, such as smoothing effects. When there are areas of high contrast or very fine detail, some artifacts can be seen. In the test results for robustness, a single image was used to compare and contrast various tests. Noise was added before and after compression, as well as flipping bits in the compressed image. These tests all showed that the code is resilient to missing data, and can still reconstruct the remainder of an image if a bit gets flipped. Different basis were also tested to see if they would work more efficiently for various types of images. Overall, it can be seen that the compression code and solver work very well in conjunction with each other. The reconstructions are all high quality with low amounts of artifacts.

[4]To perform the MPEG2 compression, we used the website *http://video.online-convert.com/convert-to-mpeg-2*.

6 Future Works

In the future, we would like to continue to improve the results on the reconstruction process, in particular, the video reconstructions. There are many techniques that involve taking advantage of structure between frames. Very little of this was used in our own method, and would be worth incorporating. We would also like to find new ways to measure the quality of the reconstructions. Though there are techniques such as SSIM or Entropy, more local measurements to measure the amount of blur or discoloring would be worth exploring as well. We would also like to perform more analysis in the compressed space and work on ways to extract specific pieces of information out of edges such as edges or regions of certain color. Lastly, we would like to extend this method to other forms of information such as audio data or data from sensors.

References

1. C. Chen, E. W. Tramel, and J. E. Fowler, *Compressed-Sensing Recovery of Images and Video Using Multihypothesis Predictions*, in Proceedings of the 45th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, November 2011, pp. 1193-1198.

2. K. Dabov, A. Foi, V. Katkovnic and K. Egiazarian, *Image Denoising by Sparse 3D Transform Collaborative Filtering*,IEEE Transactions of Image Processing, Vol. 16, No.8, Aug. 2007.

3. Richard Baraniuk, Mark A. Davenport, Marco F. Duarte, Chinmay Hegde, Jason Laska, Mona Sheikh, Wotao Yin, *An Introduction to Compressive Sensing*, Connexions, Rice University, Houston, Texas, Online: $< http://cnx.org/content/col11133/1.5/ >$.

4. S. Mun and J. E. Fowler, *Block Compressed Sensing of Images Using Directional Transforms*, Proceedings of the International Conference on Image Processing, Cairo, Egypt, November 2009, pp. 3021-3024.

5. S. Mun and J. E. Fowler, *Residual Reconstruction for Block-Based Compressed Sensing of Video*, Proceedings of the IEEE Data Compression Conference, J. A. Storer and M. W. Marcellin, Eds., Snowbird, UT, March 2011, pp. 183-192.

6. C. Chen, E. W. Tramel, and J. E. Fowler, *Compressed-Sensing Recovery of Images and Video Using Multihypothesis Predictions*, Proceedings of the 45th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, November 2011, pp. 1193-1198.

7. J. Prades-Nebot, Y. Ma, and T. Huang, *Distributed video coding using compressive sampling*, in Proceedings of the Picture Coding Symposium, Chicago, IL, May, 2009.

8. . Ward, Compressed sensing with cross validation, IEEE Transactions on Information Theory , vol. 55, no. 11, pp. 57735782, December 2009.

9. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, *Image quality assessment: From error visibility to structural similarity*, IEEE Transactions on Image Processing , vol. 13, no. 4, pp. 600612, April 2004

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 12 - 2015	Contractor Report	09/2014 - 09/2015

4. TITLE AND SUBTITLE

COxSwAIN: Compressive Sensing for Advanced Imaging and Navigation

5a. CONTRACT NUMBER

NNX13AR50A

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Kurwitz, Richard; Pulley, Marina; LaFerney, Nathan; Munoz Carlos

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

579853.04.02.08.02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, Virginia 23681

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA/CR-2015-219000

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified
Subject Category 59
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES

Final Report

Langley Technical Monitor:Michael D. Vanek

14. ABSTRACT

The COxSwAIN project focuses on building an image and video compres-sion scheme that can be implemented in a small or low-power satellite. To do this, we used Compressive Sensing, where the compression is per-formed by matrix multiplications on the satellite and reconstructed on the ground. Our paper explains our methodology and demonstrates the results of the scheme, being able to achieve high quality image compres-sion that is robust to noise and corruption.

15. SUBJECT TERMS

Image compression

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	23	(757) 864-9658