UNIVERSITY OF CAMBRIDGE

Compressive Sensing in Video Encoding

Brian Azizi

Project Progress Report

Supervised by:
Dr Anita Faul

Department of Physics
Cavendish Laboratory
JJ Thomson Avenue
CB3 0HE
Cambridge, UK

May 9, 2016

Contents

Chapter 1

Introduction

This report summarizes the progress that has been made on the main MPhil project and gives a breakdown of what work still needs to be done.

Project Outcome

The project outcome shall be a generic compressive sensing algorithm which takes as input 3D data which can be interpreted as video and encodes this efficiently and also reconstructs data when data is missing.

Progress so far

Work on the project began at the start of April. The first two and a half weeks were used to get familiar with Sparse Bayesian Learning, Wavelet Transforms, the Multi-Scale Resolution Analysis of Haar wavelets, and the Compressive Sensing framework. This involved rough Matlab implementations of the RVM algorithm and of discrete Haar wavelet transforms of 2-dimensional signals (images).

Following that, I spent roughly one week reviewing the code base for the Multi-Scale Cascade of Estimations algorithm by Georgios Pilikos [1].

Finally, for the last week and a half, I have been working on familiarizing myself with video data and on extending the signal reconstruction code from 2D signals to 3D signals (such as videos).

This progress report contains a brief summary of the relevant theory, followed by a description of the current implementation of the 3D signal reconstruction algorithm and ends with a quick breakdown of the remaining work.

Chapter 2

Theory

This chapter briefly summarizes the main theoretical building blocks of the project.

There are three parts: A signal processing framework called *Compressive Sensing (CS)*, a pre-processing step in form of a basis transformation based on discrete wavelet transforms and a Machine Learning algorithm called *Sparse Bayesian Learning.*

2.1 Compressive Sensing

This section is based on [1]. The problem to be solved can be formulated as follows: Let $\boldsymbol{x} \in \mathbb{R}^N$ be a signal of interest. We do not measure $\boldsymbol{x}$ directly and it is thus unknown. Instead, we have a measurement $\boldsymbol{y} \in \mathbb{R}^M$, with $M << N$, from which we want to reconstruct $\boldsymbol{x}$. The signals $\boldsymbol{x}$ and $\boldsymbol{y}$ are related as follows:

$$\boldsymbol{\Omega x} = \boldsymbol{y} \tag{2.1}$$

where $\boldsymbol{\Omega}$ is a known $M \times N$ matrix referred to as the *sensing matrix.*

For example, in [1], the signal of interest $\boldsymbol{x}$ is an image, so that N is equal to the total number of pixels in the image and x_i is equal to the intensity of the corresponding pixel. However, we imagine that we have only access to a corrupted version of $\boldsymbol{x}$ in which random pixel values have been deleted. This is our measurement $\boldsymbol{y}$. See Figure 2.1 for an example. The sensing matrix $\boldsymbol{\Omega}$ corresponding to this scenario is obtained by taking the $N \times N$ identity matrix and deleting the rows that correspond to the missing entries in $\boldsymbol{x}$.

Compressive Sensing (CS) is a collection of signal processing techniques that allow for efficient *reconstruction* (and indeed *aquisition*) of such signals by solving the underdetermined system (2.1).

Of course, there are infinitely many solutions to an underdetermined system. In the CS framework, we seek to find a solution $\hat{\boldsymbol{x}}$ that is *sparsest in some domain.* By that, we mean that we want to find $\hat{\boldsymbol{x}}$ that satisfies (2.1), such that

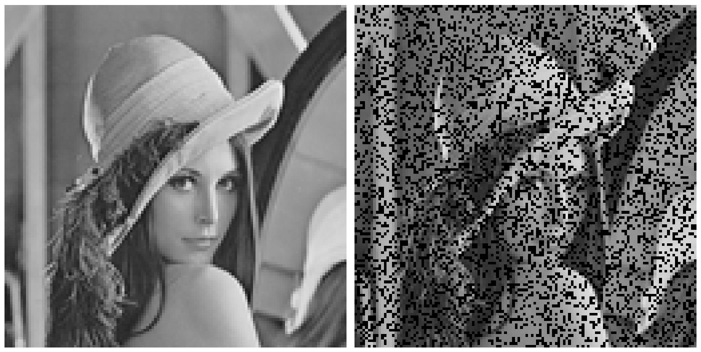


Figure 2.1: Example of a signal pair $\boldsymbol{x}$ (left) and $\boldsymbol{y}$ (right). We wish to reconstruct $\boldsymbol{x}$ from $\boldsymbol{y}$.

there exists a basis transformation of $\hat{\boldsymbol{x}}$ in which it has the smallest number of nonzero entries.

More concretely, we assume there exists a domain in which the desired signal $\boldsymbol{x}$ is sparse. I.e. there exists a $N \times N$ basis matrix Ψ such that $\boldsymbol{x} = \boldsymbol{\Psi w}$ and $\boldsymbol{w}$ is sparse.

The CS problem can then be expressed as follows:

$$\min ||\boldsymbol{w}||_0 \qquad \text{subject to} \qquad \boldsymbol{\Omega \Psi w} = \boldsymbol{y} \tag{2.2}$$

where $||.||$ denotes the l_0 norm, i.e. the number of nonzero components.

For a more detailed review of the CS framework, see [2].

2.2 Wavelets Basis Transformation

The second building block is the basis transformation

$$\boldsymbol{x} = \boldsymbol{\Psi w} \tag{2.3}$$

that takes the dense signal $\boldsymbol{x}$ and sends it into a domain in which its transformation $\boldsymbol{w} = \boldsymbol{\Psi}^T \boldsymbol{x}$ is sparse.

Finding a set of basis functions $\boldsymbol{\Psi}$ that achieve such a transformation lies at the heart of many lossy compression techniques. For instance, in image processing the JPEG 2000 standard is a widely used lossy compression technique that relies on this principle. The original image $\boldsymbol{x}$ is transformed into $\boldsymbol{w}$ using the so-called *Discrete Cosine Transform.* The basis matrix $\boldsymbol{\Psi}$ is orthogonal, so $\boldsymbol{x}$ and $\boldsymbol{w}$ have the same l_2 norm. In the original signal $\boldsymbol{x}$ the length is spread across many of its coefficients. On the other hand, most of the length of $\boldsymbol{w}$ is concentrated in a few of its coefficients. A large fraction of the entries in $\boldsymbol{w}$ are very close to zero. By deleting these entries in $\boldsymbol{w}$ and only storing the non-zero coefficients (and the corresponding basis functions), we can obtain a compressed version $\hat{\boldsymbol{w}}$. This allows us to significantly reduce the amount of data

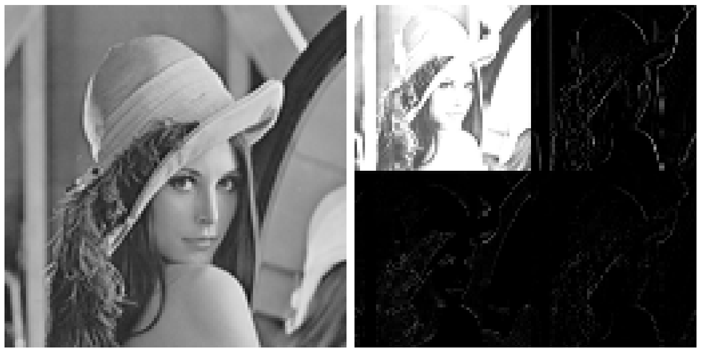


Figure 2.2: Original image $\boldsymbol{x}$ (left) and its Haar basis transformation $\boldsymbol{w}$ (right). See next chapter for more details on Haar wavelets.

that needs to be stored without affecting the visual quality in the reconstructed image $\hat{\boldsymbol{x}} = \boldsymbol{\Psi}\hat{\boldsymbol{w}}$.

It is important to note here that the choice of basis functions $\boldsymbol{\Psi}$ typically has a significant effect on the performance of the reconstruction algorithms.

The simplest wavelet basis transformation is based on the Haar wavelets. Figure 2.2 vizualises a basis transformation of the original image $\boldsymbol{x}$ to $\boldsymbol{w}$. This example uses a Haar wavelet basis at the first scale. We will explain the Haar wavelet transformations of images and videos in more detail in the next chapter. Dark areas correspond to small coefficients. Note that most entries in $\boldsymbol{w}$ are near zero. In practice, we approximate these entries as zero and treat $\boldsymbol{w}$ as sparse.

For a deeper introduction into wavelets see [3]. For more information on wavelet compression techniques, see [4].

2.3 Sparse Bayesian Learning

So far, we have not addressed the central question: How do we solve the compressive sensing problem (2.2)? Various deterministic approaches have been developed in recent years. See [1] for an overview.

In the MPhil project, we will employ a probabilistic technique based on Sparse Bayesian Learning. In particular, we will use the *Relevance Vector Machine (RVM)* [5, 6] to reconstruct $\boldsymbol{w}$ from the measurements $\boldsymbol{y}$. Following that, we obtain a reconstructed version of the desired signal $\boldsymbol{x}$ by pre-multiplying $\boldsymbol{w}$ by $\boldsymbol{\Psi}$ to obtain the desired signal.

The RVM is a *regression* technique. We model the relationship between $\boldsymbol{y}$ and $\boldsymbol{x}$ by

$$y_i = \boldsymbol{w}^T \boldsymbol{\psi}(x_i) + \epsilon_i \tag{2.4}$$

where $\psi(x_i)$ is the ith column of Ψ and the ϵ_i are independent noise variables drawn from a zero-mean Gaussian distribution $\mathcal{N}(o, \sigma^2)$.

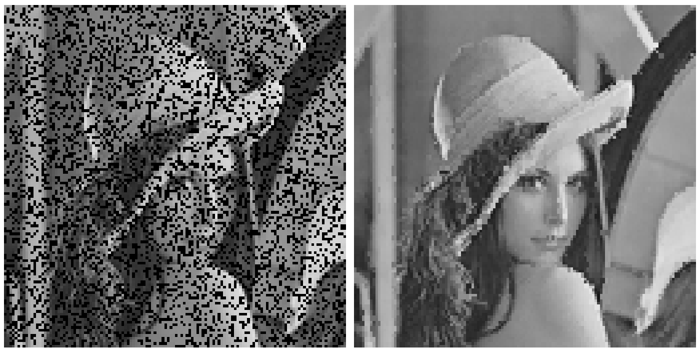


Figure 2.3: Corrupted signal $\boldsymbol{y}$ (left) and reconstructed signal $\hat{\boldsymbol{x}}$ (right) using a cascade of 3 RVMs with Haar basis functions (see [1]).

The model uses the available measurements $\{(x_1, y_1), \ldots, (x_M, y_M)\}$ as training data to train the model in a Bayesian framework. The result of training the RVM is a posterior distribution for the unknown vector of coefficients $\boldsymbol{w}$. A special feature of the RVM is that the posterior mean $\boldsymbol{\mu}$ of $\boldsymbol{w}$ is often very sparse.

In order to reconstruct the image, we use the estimated posterior mean to "predict" what a pixel value y^* should be at a location x^* in which information was missing:

$$y^* = \boldsymbol{w}^T \boldsymbol{\psi}(x^*) \tag{2.5}$$

Apart from achieving sparse solutions, one further desirable feature of the RVM is that the model provides error bars for its predictions. This is used in [1] to construct a multi-scale cascade of RVM estimations and achieve significant performance boosts.

An example of this can be seen in Figure 2.3.

For details on the RVM and its implementation see [1, 6, 5].

Chapter 3

Current Implementation

This chapter gives a brief description of the current state of our implementation of the 3D signal reconstructer.

3.1 Code Review

Our implementation is based on the work in [1]. As such, approximately one week was spent reviewing his code base in order to become familiar with code and also to look for possible bugs. Although two minor bugs were discovered, fixing them had no qualitative effect on the output of the code.

3.2 Haar basis functions

The RVM takes as input a target vector ($\boldsymbol{y}$) and a basis matrix ($\boldsymbol{\Psi}$). In this respect, it is agnostic about whether the signal is an image or video or of some other type alltogether. Most of this information is encoded in the basis matrix $\boldsymbol{\Psi}$. It is therefore important, and often challenging, to select a good set of basis functions.

Our current implementation uses 3-dimensional Haar wavelet basis functions. I will show how the basis matrix $\boldsymbol{\Psi}$ is constructed by briefly describing how the discrete Haar wavelet transform is performed on 1D, 2D and finally on 3D signals.

3.2.1 1D Haar wavelet transform

Consider a 1-dimensional signal $\boldsymbol{s} = \{s_1, \dots, s_r\} \in \mathbb{R}^r$ (r for "rows"), where, for simplicity, we assume that r is a power of 2. The Haar wavelet transform can be performed at various resolution scales. The transform at the first scale is given by:

$$\boldsymbol{s} = \{s_1, \dots, s_r\} \rightarrow \frac{1}{\sqrt{2}}\{s_1{+}s_2, s_3{+}s_4, \dots, s_{r-1}{+}s_r, s_1{-}s_2, \dots, s_{r-1}{-}s_r\} = \hat{\boldsymbol{s}}^{(1)}$$

The first half of the signal is replaced by scaled averages of adjacent elements and the second half is replaced by scaled differences of adjacent elements. By performing this transform again on the first half of $\hat{\boldsymbol{s}}^{(1)}$ while keeping the second half fixed, we get the Haar wavelet transform at the second scale $\hat{\boldsymbol{s}}^{(2)}$. To get the third scale transform $\hat{\boldsymbol{s}}^{(3)}$, we perform the initial transform on the first quarter of $\hat{\boldsymbol{s}}^{(2)}$ while keeping the rest of the signal fixed. We may continue this process until we reach the ith scale, where $2^i = r$.

From here on, we will only consider the first scale transform $\hat{\boldsymbol{s}}^{(1)}$ and we will omit the (1) superscript. We can express the transform as a multiplication by an orthogonal $r \times r$ matrix W given by

$$W = \begin{bmatrix} \Phi_r \\ \Psi_r \end{bmatrix} \tag{3.1}$$

where Φ_r and Ψ_r are $(r/2) \times r$ matrices[1] given by

$$\Phi_r = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix}$$

and

$$\Psi_r = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 \end{pmatrix}$$

In the signal processing literature, Φ_r is referred to as a low pass filter, while Ψ_r is referred to as a high pass filter. Φ_r outputs an average of the signal and Ψ_r outputs the details of the signal.

3.2.2 2D Haar wavelet transform

Let $A \in \mathbb{R}^{r \times c}$ be a 2-dimensional signal (e.g. an image). For simplicity, we will assume that both r and c are powers of 2 (though not necessarily equal).

It is simple to obtain A's Haar wavelet transform $\hat{A}$ at the first scale. This is done by first applying the 1-dimensional transform individually to each column of A to obtain a temporary matrix $\hat{A}_{temp}$. Next, we apply the 1-dimensional haar wavelet transform individually to each row of $\hat{A}_{temp}$ to obtain $\hat{A}$.

We can again express the transform as a multiplication of matrices:

$$\hat{A} = \begin{bmatrix} \Phi_r \\ \Psi_r \end{bmatrix} A \begin{bmatrix} \Phi_c^T & \Psi_c^T \end{bmatrix} \tag{3.2}$$

where Φ_r and Ψ_r are as before and Φ_c and Ψ_c are of similar form but each have dimensions $(c/2) \times c$. This is the transform that was used to generate the

[1]Note that the matrix Ψ_r used here is different to the matrix Ψ that was used in the previous chapter (which corresponds to W^T here).

RHS of Figure 2.2. We note that the high-pass filters essentially detect edges of various orientations in the image.

However, as it currently stands, we cannot use this form of the basis transformation for the reconstruction algorithm. Recall that the RVM requires a *vector* of measurements as opposed to a matrix and also that it requires a single basis matrix, not a basis transform as given in (3.2).

To do this, we store the 2-dimensional signal A as a long column vector $\boldsymbol{a}$ of length rc by pasting the individual columns of A one after another. The basis transformation of $\boldsymbol{a}$ can then be expressed as

$$\hat{\boldsymbol{a}} = W\boldsymbol{a}$$

where W is a $rc \times rc$ matrix given by

$$W = \begin{bmatrix} \Phi_c \otimes \Phi_r \\ \Phi_c \otimes \Psi_r \\ \Psi_c \otimes \Phi_r \\ \Psi_c \otimes \Psi_r \end{bmatrix}$$

The symbol $\otimes$ denotes the *Kronecker product.* The kronecker product $P \otimes Q$ between matrices P and Q with dimensions $m_P \times n_P$ and $m_Q \times n_Q$, respectively, is defined to be the block matrix

$$\begin{bmatrix} p_{1,1}Q & p_{1,2}Q & \cdots & p_{1,n_P}Q \\ p_{2,1}Q & p_{2,2}Q & \cdots & p_{2,n_P}Q \\ \vdots & \vdots & \ddots & \vdots \\ p_{m_P,1}Q & p_{m_P,2}Q & \cdots & p_{m_P,n_P}Q \end{bmatrix}$$

of size $m_P m_Q \times n_P n_Q$.

3.2.3 3D Haar wavelet transform

Let $V \in \mathbb{R}^{r \times c \times s}$ be a 3-dimensional signal such as a video. V has r rows, c columns and s slices, and we assume that r, c and s are all powers of 2. We may visualize V as a "volume" with 2 spacial dimensions and one time dimension corresponding to frames of the video.

To obtain the Haar wavelet transform $\hat{V}$ of V, we first perform the 1-dimensional transform individually on each column in every slice of V to get $\hat{V}_{temp1}$. We then perform the 1D transform on every row in every slice of $\hat{V}_{temp_1}$ to get $\hat{V}_{temp_2}$. Finally, we perform the 1D transform across the slices for every row and column to get $\hat{V}$.

However, like in the 2-dimensional case, we need to be able to pass a single vector of coefficients and a single basis matrix to the RVM. To do this, we vectorize V as follows. First, we vectorize each individual slice of V as before in the 2D case. Then, we stack all these vectors on top each other to get one very long column vector $\boldsymbol{v}$ of length rcs. The Haar wavelet transform is given by

$$\hat{\boldsymbol{v}} = W\boldsymbol{v}$$

where

$$W = \begin{bmatrix} \Phi_s \otimes \Phi_c \otimes \Phi_r \\ \Phi_s \otimes \Phi_c \otimes \Psi_r \\ \Phi_s \otimes \Psi_c \otimes \Phi_r \\ \Phi_s \otimes \Psi_c \otimes \Psi_r \\ \Psi_s \otimes \Phi_c \otimes \Phi_r \\ \Psi_s \otimes \Phi_c \otimes \Psi_r \\ \Psi_s \otimes \Psi_c \otimes \Phi_r \\ \Psi_s \otimes \Psi_c \otimes \Psi_r \end{bmatrix}$$

Comparing notation to the previous chapter, what we refer to as W here is the transpose of what was previously denoted as Ψ. And since $\boldsymbol{v} = W^T\hat{\boldsymbol{v}}$, we see that $\boldsymbol{v}$ corresponds to what was previously called $\boldsymbol{x}$.

3.3 Preliminary Results from first working Prototype

We have obtained some results with our current implementation. The implementation uses the Haar wavelet transform at the first scale.

Our example video has a resolution of 128 by 128 pixels and consists of a total of 64 frames. Thus, $r = 128$, $c = 128$ and $s = 64$. Note that even for such a relatively small sample, the size of the basis matrix Ψ is $(128 * 128 * 64) \times (128 * 128 * 64) = 1048576 \times 1048576$. Even in single precision, storing this matrix would require around 4 terrabytes.

For this reason, we have split the original input signal into $8 \times 8 \times 8$ blocks and perform the algorithm on the individual blocks.

In Figures 3.1 and 3.2, we have included a sample frame from the corrupted test video and the same frame after reconstruction.

In Figure 3.1, we corrupted the video by deleting 30% of the pixel values in the first frame and deleting the same pixel values in each subsequent frame (so the same pixels are missing in each frame). Figure 3.2 uses the same corruption scheme but we deleted 50% rather than 30% of pixel values.

These initial results are promising, though clearly there are still improvements to be made.

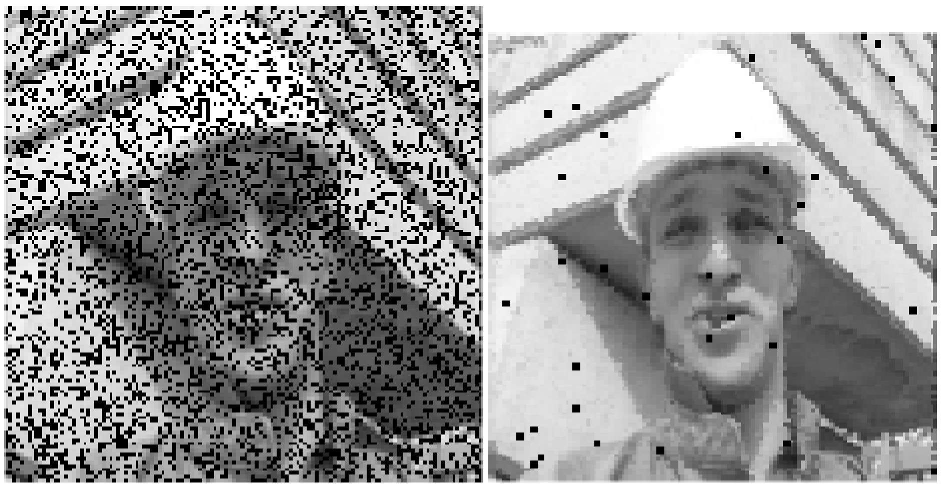


Figure 3.1: Sample frame from corrupted video (left) and the reconstructed video (right)

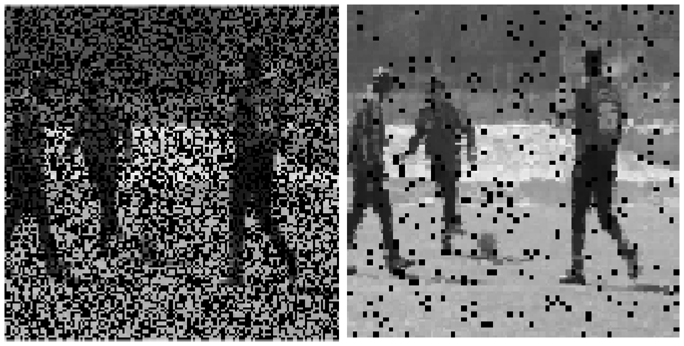


Figure 3.2: Sample frame from corrupted video (left) and the reconstructed video (right)

Chapter 4

Still To Do

4.1 Software Development

Multi-Scale Haar wavelets

Currently, the code uses Haar basis functions at the first scale. Note how in the reconstructed videos in Figures 3.3 and 3.3, there are still missing (black) patches. Using Haar basis functions at higher scales allows us to fill these missing areas.

Cascade of Estimations

Using higher scales of the Haar wavelets often results in somewhat blurry reconstructions. The image reconstruction algorithm in [1] employs a cascade of estimations. By using Haar wavelets at multiple scales in a cascade, we are able to achieve sharper reconstructions while also filling all the missing pixel values and frames.

The cascade will be the next addition to the current prototype.

Discrete Cosine Transform

Once the Haar wavelets and the cascade have been implemented, I would like to explore the use of the Discrete Cosine Transform as a choice of basis transformation. There are potentially quite significant performance gains.

Strengthening the code base

It may be worth spending some time in making the code base more robust. For example by moving the code to an object-oriented design or at least by adding a debug mode and some unit tests.

Parallel Implementation

The signal is typically broken into distinct patches which get processed independently of one another. Parallelizing the code (e.g. by using MPI or CUDA) would likely to significant speed-ups.

4.2 Dissertation

Extensive Literature Review

A significant part of the project will be an extensive review on the current literature in the field of video reconstruction and video encoding. A more extensive review of the Compressive Sensing framework may also be included here as well as, potentially, the theory of wavelets.

Performance Analysis

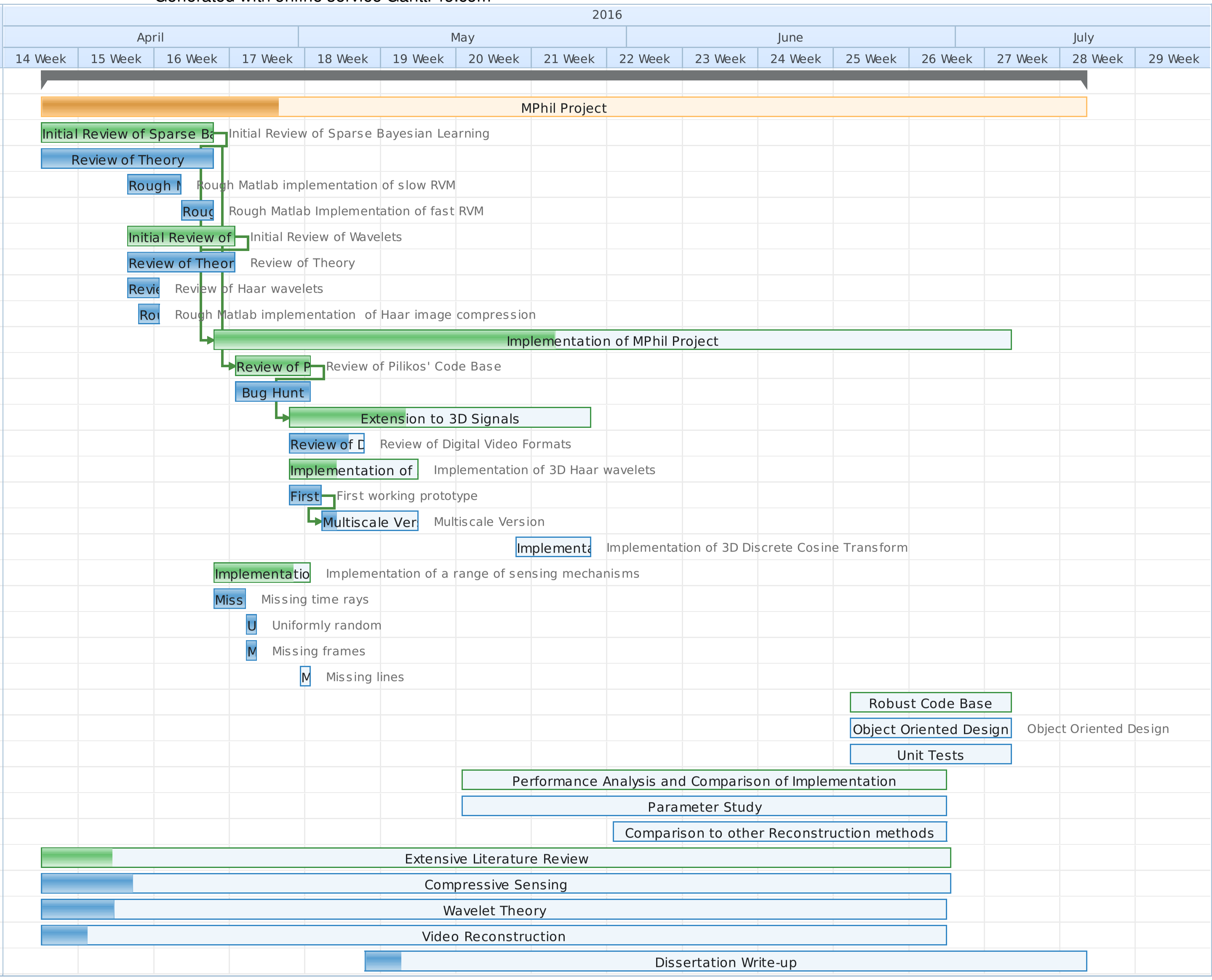
There are two parts to this. First, how is the performance affected by different settings of the parameters of the algorithm. Secondly, how does our method compare to other existing signal reconstruction methods? Under what circumstances we get superior performance? How could we improve the method?

4.3 Schedule

Having implemented a first working prototype, I believe we are well on schedule to finish on time. The plan is to have a first draft ready by mid July. The next page gives a rough outline of the schedule in form of a Gantt chart.

Generated with online service GanttPro.com

	Task name	Start date	End date	Progre...
	Total estimate	07/04/16	13/07/16	
1	MPhil Project	07/04/16	13/07/16	22%
1.1	Initial Review of Sparse Baye	07/04/16	23/04/16	100%
1.1.1	Review of Theory	07/04/16	23/04/16	100%
1.1.2	Rough Matlab implementa	15/04/16	20/04/16	100%
1.1.3	Rough Matlab Implementa	20/04/16	23/04/16	100%
1.2	Initial Review of Wavelets	15/04/16	25/04/16	100%
1.2.1	Review of Theory	15/04/16	25/04/16	100%
1.2.2	Review of Haar wavelets	15/04/16	18/04/16	100%
1.2.3	Rough Matlab implementa	16/04/16	18/04/16	100%
1.3	Implementation of MPhil Proj	23/04/16	06/07/16	42%
1.3.1	Review of Pilikos' Code Ba	25/04/16	02/05/16	100%
1.3.1.1	Bug Hunt	25/04/16	02/05/16	100%
1.3.2	Extension to 3D Signals	30/04/16	28/05/16	38%
1.3.2.1	Review of Digital Video	30/04/16	07/05/16	80%
1.3.2.2	Implementation of 3D H	30/04/16	12/05/16	36%
1.3.2.2.1	First working prototy	30/04/16	03/05/16	100%
1.3.2.2.2	Multiscale Version	03/05/16	12/05/16	15%
1.3.2.3	Implementation of 3D D	21/05/16	28/05/16	0%
1.3.3	Implementation of a range	23/04/16	02/05/16	83%
1.3.3.1	Missing time rays	23/04/16	26/04/16	100%
1.3.3.2	Uniformly random	26/04/16	27/04/16	100%
1.3.3.3	Missing frames	26/04/16	27/04/16	100%
1.3.3.4	Missing lines	01/05/16	02/05/16	0%
1.3.4	Robust Code Base	21/06/16	06/07/16	0%
1.3.4.1	Object Oriented Design	21/06/16	06/07/16	0%
1.3.4.2	Unit Tests	21/06/16	06/07/16	0%
1.4	Performance Analysis and C	16/05/16	30/06/16	0%
1.4.1	Parameter Study	16/05/16	30/06/16	0%
1.4.2	Comparison to other Reco	30/05/16	30/06/16	0%
1.5	Extensive Literature Review	07/04/16	30/06/16	7%
1.5.1	Compressive Sensing	07/04/16	30/06/16	10%
1.5.2	Wavelet Theory	07/04/16	30/06/16	8%
1.5.3	Video Reconstruction	07/04/16	30/06/16	5%
1.6	Dissertation Write-up	07/05/16	13/07/16	5%



Bibliography

[1] G. Pilikos, Signal reconstruction using compressive sensing, MPhil thesis, University of Cambridge (2014).

[2] E. J. Candès, M. B. Wakin, An introduction to compressive sampling, Signal Processing Magazine, IEEE 25 (2) (2008) 21–30.

[3] E. J. Stollnitz, T. D. DeRose, D. H. Salesin, Wavelets for computer graphics: a primer. 1, Computer Graphics and Applications, IEEE 15 (3) (1995) 76–84.

[4] R. A. DeVore, B. Jawerth, B. J. Lucier, Image compression through wavelet transform coding, Information Theory, IEEE Transactions on 38 (2) (1992) 719–746.

[5] M. E. Tipping, Sparse bayesian learning and the relevance vector machine, The journal of machine learning research 1 (2001) 211–244.

[6] M. E. Tipping, A. C. Faul, et al., Fast marginal likelihood maximisation for sparse bayesian models., in: AISTATS, 2003.