2016

Selective Sampling for Compression and Effective Reconstruction

Colin F. Merrick
University of Rhode Island, merrick_colin@hotmail.com

Follow this and additional works at: <https://digitalcommons.uri.edu/theses>

Recommended Citation

Merrick, Colin F., "Selective Sampling for Compression and Effective Reconstruction" (2016). *Open Access Master's Theses*. Paper 922.
<https://digitalcommons.uri.edu/theses/922>

This Thesis is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Master's Theses by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons-group@uri.edu.

SELECTIVE SAMPLING FOR COMPRESSION AND EFFECTIVE
RECONSTRUCTION

BY

COLIN F. MERRICK

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2016

MASTER OF SCIENCE THESIS
OF
COLIN F. MERRICK

APPROVED:

Thesis Committee:

Major Professor Ashwin Sarma

Richard J. Vaccaro

Tom Bella

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2016

ABSTRACT

The current standard when creating a digital image is to have a grid of sensors with each sensor collecting information that defines its associated pixel value in a digital image. For many applications storing or transmitting the entire set of pixel values is not desirable due to its large size. In these cases processing is performed on the original data which reduces its size, but results in an image which is an approximation of the original. We present an alternative to this approach that will only use a subset of the existing sensors, thus collecting less initial data and therefore avoiding the need for the additional step of having to reduce the data size after sampling. For this approach, which uses sub-sampling, we define an image construction method which takes the sub-samples and produces its own approximation of the fully sampled image. With this construction method and a strict constraint on the kinds of images we are allowed to capture, we show that there exists an optimal set of pixel locations to sample to minimize the squared error between the constructed image and the fully sampled image. Since our method has less information about the image when it decides how to reduce its size, when compared to the current standard, our method will result in a less accurate approximation of the original image. Even with this lowered performance we will show that using our supplied construction method and sample locations under the specified constraint on the original image, the proposed method creates an image that is a visually acceptable approximation.

ACKNOWLEDGMENTS

I would like to convey my appreciation to my thesis advisor Professor Ashwin Sarma for his guidance during every step of the thesis process and for his never ending patience. Also I would like to thank Amanda Callahan for proofreading many iterations of this document. Finally I would like to acknowledge the financial support of Department of the Navy's Acquisition Workforce Tuition Assistance Program and the Naval Undersea Warfare Center Division Newport training office.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGMENTS	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	v
CHAPTER	
1 Introduction	1
2 Inspiration and Background Material	4
3 Image Approximation	11
4 Image Constraint, Construction Method, and Optimal Sample Locations	16
5 Selective Sampling Results	21
6 Selective Sampling Conclusion	35
LIST OF REFERENCES	37
BIBLIOGRAPHY	38

LIST OF FIGURES

Figure		Page
1	Barbara and sub-images	1
2	Barbara sub-image numerical representation	2
3	Image partition	4
4	The DCT2 of a sub-image	5
5	The DCT2 basis	5
6	Zigzag ordering	6
7	The DCT1 basis	7
8	DCT2 image created by DCT1 vectors	8
9	Mean energy over a set of images	12
10	Mean zigzag energy over a set of images	12
11	Sparse Barbara images	13
12	Sparse Barbara sub-images	14
13	Linear combination of images	17
14	$(s,n)=(2,1)$, optimal sample locations	22
15	$(s,n)=(3,2)$, optimal sample locations	22
16	$(s,n)=(4,3)$, optimal sample locations	22
17	$(s,n)=(5,4)$, optimal sample locations	23
18	$(s,n)=(6,5)$, optimal sample locations	23
19	$(s,n)=(7,6)$, optimal sample locations	23
20	$(s,n)=(8,7)$, optimal sample locations	23
21	n equals 1, image approximation	24

Figure		Page
22	n equals 2, image approximation	25
23	n equals 3, image approximation	26
24	n equals 4, image approximation	27
25	n equals 5, image approximation	28
26	n equals 6, image approximation	29
27	n equals 7, image approximation	30
28	n equals 1, sub-image approximation	31
29	n equals 2, sub-image approximation	31
30	n equals 3, sub-image approximation	32
31	n equals 4, sub-image approximation	32
32	n equals 5, sub-image approximation	33
33	n equals 6, sub-image approximation	33
34	n equals 7, sub-image approximation	34

CHAPTER 1

Introduction

A digital image, is a picture which is made up entirely of shaded squares called pixels, whose shadings are selected from a finite set. In figure 1, a zoomed in view of the commonly used test image, Barbara, shows that indeed the image is made up of shaded pixels.

Figure 1: Barbara and sub-images

To create the Barbara image using a digital camera, there is a sensor in the camera dedicated to each pixel in the image. Each pixel's shading would be stored as a number. As an example of this the most zoomed in portion in figure 1 could also be viewed as a grid of numbers whose entries represent their associated pixel shading as shown in figure 2.

Saving all of these pixel values is often referred to as the raw image. The problem with storing the raw image is that it takes up a lot of memory. To counter this issue the raw image is encoded in such a way that its total memory footprint is

151	146	142	137	135	131	129	125
117	110	109	98	95	89	83	77
73	67	62	54	54	50	49	46
51	52	52	55	63	67	74	84
93	103	103	111	120	122	132	137
148	156	153	159	163	165	165	168
167	174	168	170	172	169	167	163
160	160	157	154	147	145	137	132

Figure 2: Barbara sub-image numerical representation

greatly reduced, but this comes at the expense of the image, meaning once this new data is decoded the resulting image is no longer the same as the raw version, rather it is a good approximation. This is known as lossy compression. The progression of taking a full grid of samples and then performing lossy compression is the typical way in which digital images are created and then compressed for storage. This chain of events will be referred to as complete sampling followed by compression.

Before moving on it is worth stating that many of the details of digital cameras, digital images, and image compression will be greatly simplified or ignored. For example we will only be working with gray scale images in this paper, this will all be done to avoid unnecessary complexity.

In this paper we will present an alternative to complete sampling followed by compression. We will propose taking a reduced set of samples, and by doing so, our data will be smaller than the raw image and thus will not need the step of lossy compression, while also requiring less sensors. Since we will have only collected some of the pixel values we will need a construction method that takes the pixel values we do have and estimates the others we do not.

We will place a strict constraint on the types of images we are allowed to sample which will be described later. With the construction method and the constrained image we will then show that there is an optimal set of pixel locations to sample such that the total squared error between our constructed image and the raw image is minimized.

In chapter 2, we will present some background material on a popular lossy compression method that served as a motivation for the proposed method. In chapter 3, we will show how a certain kind of image approximation technique yields visually pleasing results, and as such will be leveraged in our image construction method. Then in chapter 4 we will layout our image constraint, our construction method, and show that there are better locations to sample than others to minimize the squared error between the would-be raw image and our constructed image. In chapter 5 we present some result of applying the presented approach and finally in chapter 6 give a conclusion.

CHAPTER 2

Inspiration and Background Material

The framework for our method is heavily inspired by a very popular image compression standard known as sequential Joint Photographic Experts Group (JPEG). For the purpose of context and to lay the ground work for what is to come we will step through a high level description of JPEG and some of its components [1][2].

The JPEG standard for compression has five major components, three of which we will utilize in some respect in our proposed method. The first of the five being JPEG partitions a raw image into sub-images; the rest of the JPEG processes operate on each sub-image independent of the others. A common sub-image size is 8 pixels by 8 pixels. In figure 3 we show an example of a 16 by 16 pixel image being partitioned into 8 by 8 sub-images. Secondly, JPEG takes the

Figure 3: Image partition

2D Discrete Cosine Transform (DCT2)[3] of each sub-image. The DCT2 operation transforms the numerical representation of a sub-image into another numerical

representation of the sub-image, but now in the DCT2 domain. An example of this is shown in figure 4. What the DCT2 is doing is taking the numerical sub-

Figure 4: The DCT2 of a sub-image

image and representing it as a linear combination of a predefined set of numerical images which make up the DCT2 basis. This basis consists of 8 by 8 images which span the entire 8 by 8 space. In figure 5 we show the 64 images that define the DCT2 basis. We have put them in a grid format to show what scalar multiple, coefficient, in the DCT2 domain from figure 4, each is associated with.

Figure 5: The DCT2 basis

The third step of uniquely scaling down each DCT2 coefficient is performed followed by the fourth step of rounding each scaled coefficient toward zero. These two steps are of little importance in relation to this paper, but are necessary to

mention as steps in the JPEG process. The fifth step is the ordering of the rounded scaled coefficients. JPEG uses the zigzag ordering shown in figure 6. This ordering

Figure 6: Zigzag ordering

of the rounded scaled DCT2 coefficients tends to create long runs of zeros at the tail end of the indexing. This is desirable for further compression.

Once every sub-image has undergone JPEG compression it has been greatly reduced in size and is easy to store or transmit. At some point we will wish to view the image and will then need to decompress the image. JPEG decompression will undo every step in reverse order with the exception of the round toward zero step as it is non invertible. The result will be an image, though this image is not exactly the original it is usually a very close approximation. Undoing the DCT2 is performed by the 2D Inverse Discrete Cosine Transform (IDCT2). The IDCT2 scales each image in the DCT2 basis by its associated coefficient and then sums all the scaled images together with a pixel wise summing operation to create an image.

The DCT2 and the IDCT2 are at the core of our proposed method, so it is worth taking a finer look at these two operations. As we will show the images in the DCT2 basis are built of vectors from the 1D Discrete Cosine Transform (DCT1) basis and therefore the DCT1 is a good starting point for a proper introduction to the DCT2.

Figure 7: The DCT1 basis

The DCT1 operation takes a vector, in our case of length 8, and represents it as linear combination of discrete cosine vectors. The discrete cosine vectors used form an orthonormal basis which spans the 8 dimensional space. We show each of these row vectors, v_k along with a trace line of the scaled and shifted cosine function they were sampled from in figure 7. One can see that they consist of shifted cosine functions of increasing frequency. The formal definition of each v_k is given as follows.

$$v_k[n] = a_k \cos\left(\frac{(k-1)(n-\frac{1}{2})\pi}{8}\right) \text{ for } k, n \in \{1, 2, \dots, 8\}$$

$$a_k = \begin{cases} \frac{1}{\sqrt{8}}, & \text{if } k = 1 \\ \frac{1}{2}, & \text{if } k \neq 1 \end{cases}$$

In the DCT1 basis there is a fair amount of number repetition, out of the 64 numbers there are only 14 that are unique with 7 of the 14 being the negative of the other 7. One of the more important properties of the DCT1 basis is that it is orthonormal, meaning $v_i^T v_j$ equal 1 if $i = j$ and zero otherwise.

The DCT2 is the two dimensional version of the DCT1 and therefore has many similar properties. Instead of a basis of vectors as in the DCT1 case, the DCT2

can be intuitively visualized as a basis of images. Again all 64, 8 by 8, images in the basis, denoted $B(1), B(2), \dots, B(64)$ are shown in figure 5 with the numbering referring to its location in the zigzag ordering scheme shown in figure 6.

Each image in the DCT2 basis is the outer product of vectors from the DCT1 basis, $v_i^T v_j$. For example, the $(1, 2)$ indexed 8 by 8 matrix in figure 5, $B(2)$, is the outer product of v_1 and v_2 from figure 7, and is depicted in figure 8.

Figure 8: DCT2 image created by DCT1 vectors

With the DCT2 basis being a function of the DCT1 basis it is not that surprising that out of the 4096 numbers in the DCT2 basis only 56 are unique with 28 being the negative of the other 28 value.

To show that images in the DCT2 basis are orthonormal under element wise multiplication (.) followed by summation we will use their DCT1 outer product definition, and their orthonormal property, $v_i v_j^T = 1$ if $i = j$ and zero otherwise.

$$\begin{aligned}
B(m) &= v_w^T v_x, \quad B(n) = v_y^T v_z \\
\sum_{i=1}^8 \sum_{j=1}^8 B(m)_{i,j} \cdot B(n)_{i,j} &= \text{tr}(B(m)B(n)^T) \\
&= \text{tr}(v_w^T v_x (v_y^T v_z)^T) \\
&= \text{tr}(v_w^T v_x v_z^T v_y) \\
&= \langle v_x, v_z \rangle \text{tr}(v_w^T v_y) \\
&\text{if } x \neq z, = 0 \\
&\text{if } x = z, = \text{tr}(v_w^T v_y) \\
&= \langle v_w, v_y \rangle \\
&\text{if } w \neq y, = 0 \\
&\text{if } w = y \text{ and } x = z, = 1
\end{aligned}$$

The IDCT2 by its name undoes what the DCT2 performs. The IDCT2 takes the coefficients and scales each image in the basis accordingly and then sums them together to get the raw 8 by 8 image, R . With this definition of the IDCT2 we can represent it as a system of linear equations.

$$\begin{bmatrix} R_{1,1} \\ | \\ R_{8,8} \end{bmatrix} = \begin{bmatrix} B(1)_{1,1} & - & B(64)_{1,1} \\ | & & | \\ B(1)_{8,8} & - & B(64)_{8,8} \end{bmatrix} \begin{bmatrix} C_{1,1} \\ | \\ C_{8,8} \end{bmatrix}$$

In short, the IDCT2 operation can be performed by a matrix multiplication if the coefficients and pixel values are stacked in vectors rather than grids. We can then call the IDCT2 matrix M , the raw image R , and the coefficient matrix C , represented as a column vector, r and c respectively, to yield

$$r = Mc$$

When referring to the entries of r and c their subscripts will coincide with the numbering associated with the zigzag indexing. For example $R_{2,2}$, is equal to r_5 .

Similarly when referring to elements of B it might be more consistent to reference there entries by the zigzag ordering as well. Notice that the IDCT2 matrix is orthonormal since the DCT2 basis is orthonormal

$$\begin{bmatrix} B(1)_1 & - & B(64)_1 \\ | & & | \\ B(1)_{64} & - & B(64)_{64} \end{bmatrix} \begin{bmatrix} B(1)_1 & - & B(64)_1 \\ | & & | \\ B(1)_{64} & - & B(64)_{64} \end{bmatrix}^T = I$$

With this in hand we can show how the squared error in an estimate of the DCT2 coefficients is equal to the squared error in their image reproduction from these estimated coefficients. We will use this later when we want to show that our method minimizes the squared error in our image construction. Instead of showing this directly we will show that we minimize the squared error in the DCT2 coefficient estimates.

$$(c - \hat{c})^T (c - \hat{c})$$

$$(c - \hat{c})^T M^T M (c - \hat{c})$$

$$(r - \hat{r})^T (r - \hat{r})$$

CHAPTER 3

Image Approximation

If you have ever viewed a web page containing an image, using a slow internet connection, you may have experienced first observing a very blocky image, which as time passes becomes more detailed. If you are familiar with this effect, chances are you have viewed a progressive JPEG image [1][2]. The idea behind progressive JPEG is an approximation of the JPEG image can be sent quicker than the entire JPEG image and would provide the user with enough information about the image to decide if they wish to wait longer for the higher quality version. How this is done is quite simple. The JPEG coefficients of each sub-image, the DCT2 coefficients quantized and rounded, are not sent all at once. Rather disjoint coefficient sets for each sub-image are sent and the user recreates each sub-image using all of its received JPEG coefficients while assuming the ones not received are zero. Eventually when the full set of coefficients has been received, the complete JPEG image can be created. The question then becomes what order should the coefficients be sent such that each approximate sub-image is as good as possible. If we sent the JPEG coefficients of all sub-images in descending order of the magnitude of their associated DCT2 coefficients, then this would minimize the squared error between all of the approximate sub-images and the JPEG sub-images.

The problem with sending the JPEG coefficients with the largest associated DCT2 values for each sub-image first, is that the ordering cannot be assumed and would need to be coded into the messages. This would enlarge the data which we already decided needed to be temporarily reduced for the sake of timeliness. The compromise was to send the coefficients in the zigzag pattern, as based on empirical data the magnitudes tend to decrease with this indexing [3][4][5]. As a

quick check of this characteristic for the DCT2 we took the mean energy of all the DCT2 coefficients of every 8 by 8 sub-image on a set of Portable Gray Map (.pgm) images and have shown this for each coefficient in figure 9. The set of images used for this test consist of all the images on a Purdue Universtiy test image website [6].

$$\begin{bmatrix} 1192039 & 7868 & 2504 & 1002 & 478 & 259 & 137 & 84 \\ 7492 & 2060 & 927 & 412 & 200 & 108 & 62 & 42 \\ 2367 & 930 & 536 & 274 & 145 & 81 & 50 & 35 \\ 1056 & 445 & 295 & 180 & 103 & 63 & 43 & 30 \\ 593 & 252 & 178 & 121 & 76 & 48 & 35 & 26 \\ 330 & 150 & 114 & 82 & 56 & 42 & 34 & 25 \\ 211 & 101 & 82 & 61 & 45 & 36 & 36 & 25 \\ 138 & 86 & 73 & 57 & 43 & 36 & 29 & 22 \end{bmatrix}$$

Figure 9: Mean energy over a set of images

Next, we plotted the results ordering the energy via the zigzag pattern on a semi-log plot in figure 10.

Figure 10: Mean zigzag energy over a set of images

To get a feel for how well this kind of image approximation works we will show

Figure 11: Sparse Barbara images

the original Barbara image and then show three versions of it: the ones comprised only using the first two, six, and ten zigzag ordered coefficients from every sub image. One can see in figure 11 that between six and ten most of the information about the image is presented.

For a zoomed in look we show the same result for the upper left Barbara sub-image in figure 12.

The important takeaway is this: before we sample a sub-image there is no

Figure 12: Sparse Barbara sub-images

way to tell what ordering of the DCT2 coefficients would guarantee that their magnitudes would monotonically decrease. With this in mind one ordering that tends, but does not guarantee, to decrease over its indexing is the zigzag ordering. In the next chapter we will assume that a sub-image is made up of the first s DCT2 basis functions as indexed by the zigzag pattern, implying the other $64 - s$ are not present, or zero, in our sub-image. If we could estimate $s - 1$ of the s associated coefficients, which one should we leave out? Based on what we have just presented, it would seem the non-zero coefficient we should exclude would be the one with the largest zigzag index as it is likely to have the smallest magnitude.

CHAPTER 4

Image Constraint, Construction Method, and Optimal Sample Locations

As previously mentioned we will place a strong constraint on the kind of image we are allowed to capture. This constraint: if the raw image is taken, each 8 by 8 sub-image is s -sparse in the DCT2 domain. Meaning at most s of the DCT2 coefficients are nonzero. Further the indices of the s potential nonzero coefficients are known and the same for each sub-image. We will assume the s coefficients that might not be zero are the first s coefficients as defined by the zigzag ordering. Though we are making this assumption it is worth mentioning that with slight modification the presented method will work with any predefined set of s potential nonzero coefficients. All subsequent references to the ordering of the DCT2 coefficients, or the images in the DCT2 basis, will imply the zigzag ordering.

Now that we have our image constraint we can present our construction method. Remember we are proposing taking a reduced set of samples, thus the need for a method to estimate pixel values at locations we did not sample. The construction method will operate on each sub-image region independently. The number of samples we will be permitted to take for each sub-image region will be n which is equal to $s - 1$, giving us a compression ratio of $64:n$. For simplicity, the construction technique can be broken up into two components. The first being a DCT2 coefficient estimator, and the later, the constructor, which takes the IDCT2 of the estimated coefficients and creates a sub-image. We will present the method with an example running parallel with the general form.

Let us assume s equals 3. This is something we know a priori, so we know the raw image, R , if it were captured would be a linear combination of the first three images, $B(1), B(2), B(3)$, from the DCT2 basis. This can be seen in figure 13

with the note that the images on the left side of the equality have the same color scale while the raw image on the right side of the equality has a different color scale. This was done to show the detail in each image. We could also write what

Figure 13: Linear combination of images

is shown in figure 13 as $c_1B(1) + c_2B(2) + c_3B(3) = R$ with the c 's representing the DCT2 coefficients $c_1 = 400.1$, $c_2 = -40.5$, and $c_3 = 20.9$. If we were to take our permitted 2, or n , samples from R we could sample the pixel locations $R_{4,3}$ and $R_{4,6}$ which are equal to r_{19} and r_{41} . In this case we could write the equation for these two pixel values in linear algebraic form, with subscripts pertaining to pixel locations, as

$$\begin{bmatrix} r_{19} \\ r_{41} \end{bmatrix} = \begin{bmatrix} B(1)_{19} & B(2)_{19} \\ B(1)_{41} & B(2)_{41} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + c_3 \begin{bmatrix} B(3)_{19} \\ B(3)_{41} \end{bmatrix}$$

Or in the general case as

$$\begin{bmatrix} r_{i_1} \\ | \\ r_{i_n} \end{bmatrix} = \begin{bmatrix} B(1)_{i_1} & - & B(n)_{i_1} \\ | & & | \\ B(1)_{i_n} & - & B(n)_{i_n} \end{bmatrix} \begin{bmatrix} c_1 \\ | \\ c_n \end{bmatrix} + c_{n+1} \begin{bmatrix} B(n+1)_{i_1} \\ | \\ B(n+1)_{i_n} \end{bmatrix} \quad (1)$$

Since we are only allowed 2, or n , samples we can not solve for all three coefficients instead we will assume c_3 , or c_{n+1} , is zero and estimate the first two, or n . By assuming c_3 , or c_{n+1} , is zero. We can now represent r_{19} and r_{41} as

$$\begin{bmatrix} r_{19} \\ r_{41} \end{bmatrix} = \begin{bmatrix} B(1)_{19} & B(2)_{19} \\ B(1)_{41} & B(2)_{41} \end{bmatrix} \begin{bmatrix} \hat{c}_1 \\ \hat{c}_2 \end{bmatrix}$$

Or in general form

$$\begin{bmatrix} r_{i_1} \\ | \\ r_{i_n} \end{bmatrix} = \begin{bmatrix} B(1)_{i_1} & - & B(n)_{i_1} \\ | & & | \\ B(1)_{i_n} & - & B(n)_{i_n} \end{bmatrix} \begin{bmatrix} \hat{c}_1 \\ | \\ \hat{c}_n \end{bmatrix}$$

yielding our estimator for s equal 3 as

$$\begin{bmatrix} \hat{c}_1 \\ \hat{c}_2 \end{bmatrix} = \begin{bmatrix} B(1)_{19} & B(2)_{19} \\ B(1)_{41} & B(2)_{41} \end{bmatrix}^{-1} \begin{bmatrix} r_{19} \\ r_{41} \end{bmatrix}$$

Again in general form

$$\begin{bmatrix} \hat{c}_1 \\ | \\ \hat{c}_n \end{bmatrix} = \begin{bmatrix} B(1)_{i_1} & - & B(n)_{i_1} \\ | & & | \\ B(1)_{i_n} & - & B(n)_{i_n} \end{bmatrix}^{-1} \begin{bmatrix} r_{i_1} \\ | \\ r_{i_n} \end{bmatrix} \quad (2)$$

It is worth noting that we must pick locations to sample such that the matrix we created above is invertible or our estimator yields no results. In addition we could have picked different locations to sample and ended up with a different matrix and a different sample vector, r , and as such completely different coefficient estimates. Finally, if we have a different s value, our vectors and matrix would have been scaled in size accordingly. Now that we have finished the estimation portion of the construction technique all that would be left to do is zero fill the rest of the coefficients and take the IDCT2 giving us our approximate image.

The construction method was not designed in haste. The idea that we assume the last indexed coefficient is zero rather than any of the others was addressed at the end of the last chapter. As well the idea that we do not need all of an image's coefficients to accurately depict an image's important information was the main topic of the last chapter. With this in mind, if our estimator is able to get accurate estimates of the first n coefficients for each sub-image the complete image would seem to be a visually acceptable approximation.

Finally we arrive at the question we wanted to address: under our image constraint and using the presented construction method, are there better locations to sample such that the error in our reconstructed image can be minimized in a least squared sense? The answer is yes. As we addressed at the end of chapter 2, if we estimate the DCT2 coefficients $\hat{c}$ in column vector form, and recreate a sub-image by taking the IDCT2, the squared error in the estimated image is equal

to the squared error in the coefficient estimate. Given our image constraint and our coefficient estimator, our least squares coefficient error is

$$\begin{aligned}\epsilon^2 &= (c - \hat{c})^T (c - \hat{c}) \\ \epsilon^2 &= \left(\begin{bmatrix} c_1 \\ | \\ c_n \\ c_{n+1} \\ 0 \\ | \\ 0 \end{bmatrix} - \begin{bmatrix} \hat{c}_1 \\ | \\ \hat{c}_n \\ 0 \\ 0 \\ | \\ 0 \end{bmatrix} \right)^T \left(\begin{bmatrix} c_1 \\ | \\ c_n \\ c_{n+1} \\ 0 \\ | \\ 0 \end{bmatrix} - \begin{bmatrix} \hat{c}_1 \\ | \\ \hat{c}_n \\ 0 \\ 0 \\ | \\ 0 \end{bmatrix} \right) \\ \epsilon^2 &= c_{n+1}^2 + \left(\begin{bmatrix} c_1 \\ | \\ c_n \end{bmatrix} - \begin{bmatrix} \hat{c}_1 \\ | \\ \hat{c}_n \end{bmatrix} \right)^T \left(\begin{bmatrix} c_1 \\ | \\ c_n \end{bmatrix} - \begin{bmatrix} \hat{c}_1 \\ | \\ \hat{c}_n \end{bmatrix} \right) \quad (3)\end{aligned}$$

Since our estimation method does not allow us to attempt to estimate c_{n+1} other than to assume it is zero, the c_{n+1}^2 term will always be present and out of our control. If we solve for the c vector in equation 1 we get

$$\begin{bmatrix} c_1 \\ | \\ c_n \end{bmatrix} = \begin{bmatrix} B(1)_{i_1} & - & B(n)_{i_1} \\ | & & | \\ B(1)_{i_n} & - & B(n)_{i_n} \end{bmatrix}^{-1} \left(\begin{bmatrix} r_{i_1} \\ | \\ r_{i_n} \end{bmatrix} - c_{n+1} \begin{bmatrix} B(n+1)_{i_1} \\ | \\ B(n+1)_{i_n} \end{bmatrix} \right) \quad (4)$$

If we subtract equation 2 from equation 4 we get

$$\begin{bmatrix} c_1 \\ | \\ c_n \end{bmatrix} - \begin{bmatrix} \hat{c}_1 \\ | \\ \hat{c}_n \end{bmatrix} = -c_{n+1} \begin{bmatrix} B(1)_{i_1} & - & B(n)_{i_1} \\ | & & | \\ B(1)_{i_n} & - & B(n)_{i_n} \end{bmatrix}^{-1} \begin{bmatrix} B(n+1)_{i_1} \\ | \\ B(n+1)_{i_n} \end{bmatrix} \quad (5)$$

Substituting equation 5 into the squared error equation 3 we get a total squared error of

$$\epsilon^2 = c_{n+1}^2 + c_{n+1}^2 \left\| \begin{bmatrix} B(1)_{i_1} & - & B(n)_{i_1} \\ | & & | \\ B(1)_{i_n} & - & B(n)_{i_n} \end{bmatrix}^{-1} \begin{bmatrix} B(n+1)_{i_1} \\ | \\ B(n+1)_{i_n} \end{bmatrix} \right\|^2 \quad (6)$$

Realizing that we can not control what the value of c_{n+1}^2 is, the only part of equation 6 that we can control is

$$\epsilon_{controllable}^2 = \left\| \begin{bmatrix} B(1)_{i_1} & - & B(n)_{i_1} \\ | & & | \\ B(1)_{i_n} & - & B(n)_{i_n} \end{bmatrix}^{-1} \begin{bmatrix} B(n+1)_{i_1} \\ | \\ B(n+1)_{i_n} \end{bmatrix} \right\|^2 \quad (7)$$

All the values on the right side of the equality in equation 7 are dependent on which pixel locations we choose to sample, with some sample locations not even being valid if the matrix given is not invertible. To minimize the value of equation 7 we currently must check over all combinations of n pixel locations and look to see which combinations yields the smallest squared error. It is the case that for some values of n there are multiple combinations that minimize the squared error as will be shown in the next chapter.

CHAPTER 5

Selective Sampling Results

To find the minimum squared error for our construction method given our image constraint for a given s value we currently look at all $\binom{64}{s-1}$ combinations of pixel locations and if they are valid locations, that create an invertible matrix, we calculate the value for equation 15. Unfortunately for an s value greater than 8 this becomes extremely processing intensive and as such we have only been able to obtain results up to and including s equals 8.

Performing this brute force search seems to yield multiple sets of sample locations that tie for the title of being optimal. Below in figure 14 to figure 20 we show all combinations of optimal sample locations for s equals 2 to 8. It is interesting that for each optimal combination its vertical and horizontal reflection is also an optimal sample combination. This is not proven but rather an observation for S from 2 to 8.

Since the Barbara image does not meet our image constraint for each s value we create versions of the Barbara image that do. We do this by taking each sub-image's DCT2 and zeroing out the necessary coefficients values for a given s value then reconstruct each sub-image by taking the IDCT2. In figure 21 to figure 27 we show the constrained image followed by the one produced by one of the optimal sample sets for a given s value. In figure 28 to figure 34 we show the the same results zooming in on the upper left sub-image. In these figures it can be observed that the sub-sampled reconstructed image using the optimal sample locations creates a visual similar image to the original constrained image. This is the result we had stated we would show.

Figure 14: $(s,n)=(2,1)$, optimal sample locations

Figure 15: $(s,n)=(3,2)$, optimal sample locations

Figure 16: $(s,n)=(4,3)$, optimal sample locations

Figure 17: $(s,n)=(5,4)$, optimal sample locations

Figure 18: $(s,n)=(6,5)$, optimal sample locations

Figure 19: $(s,n)=(7,6)$, optimal sample locations

Figure 20: $(s,n)=(8,7)$, optimal sample locations

(a) 2 sparse image

(b) Image approximation

Figure 21: n equals 1, image approximation

(a) 3 sparse image

(b) Image approximation

Figure 22: n equals 2, image approximation

(a) 4 sparse image

(b) Image approximation

(a) 5 sparse image

(b) Image approximation

(a) 6 sparse image

(b) Image approximation

Figure 25: n equals 5, image approximation

(a) 7 sparse image

(b) Image approximation

(a) 8 sparse image

(b) Image approximation

(a) 2 sparse sub-image

(b) Image approximation

Figure 28: n equals 1, sub-image approximation

(a) 3 sparse sub-image

(b) Image approximation

Figure 29: n equals 2, sub-image approximation

(a) 4 sparse sub-image

(b) Image approximation

Figure 30: n equals 3, sub-image approximation

(a) 5 sparse sub-image

(b) Image approximation

Figure 31: n equals 4, sub-image approximation

(a) 6 sparse sub-image

(b) Image approximation

Figure 32: n equals 5, sub-image approximation

(a) 7 sparse sub-image

(b) Image approximation

Figure 33: n equals 6, sub-image approximation

(a) 8 sparse sub-image

(b) Image approximation

Figure 34: n equals 7, sub-image approximation

CHAPTER 6

Selective Sampling Conclusion

To recap, a brief introduction of the current standard for lossy image compression, complete sampling followed by compression, was given. It was stated that this approach requires a sensor for each pixel location in an image and needs further processing after all the sensor data is collected. Before going into any detail we stated that we would propose a compression method that would not require a sensor for every pixel in the image and by collecting less initial data would not be burdened with the need for compression processing on the sampled data.

As a segue into a more detailed understanding of the current standard and our proposed alternative we stepped through a simplified version of a popular lossy image compression method, JPEG. While going through some of the steps in the JPEG processing, we were also introducing some of the ideas we would leverage in our proposed method. The JPEG concepts of operating on sub-images independently as well as using the DCT2 and IDCT2 were all going to be exploited in the proposed alternative.

The last bit of information presented before getting into the proposed approach was that of image approximation. More specifically how using a sub-set of a sub-image's DCT2 coefficients can be used to produce good image approximations. The reason for this was to put our approach on good footing as we do not attempt to estimate all of a sub-images DCT2 coefficients. Furthermore, the image approximation method lent insight into which coefficients we decided not to attempt to estimate.

Once, in chapter 5, we laid out our image construction method and the constraint on the kinds of images we were allowed to capture and how many samples

were allowed per sub-image. Once the rules of our problem were defined, we walked through the matrix algebra of the squared error until we had an equation that was the product of a positive number we could not control and another positive number which was dependent on the sample locations. This isolated the squared error due to the pixel locations that were chosen from sampling. This is the key point in this paper, that selectively sampling certain locations can decrease the error in our image approximation. Furthermore, we found the optimal sample combinations for s equals 2 to 8.

Unfortunately, the current method for finding the set of optimal sample combinations is a brute force search which imposed limitations on the depth of our results. The observation of symmetry among optimal combinations and the fact that the DCT2 basis only consists of 56 unique numbers with only 28 unique magnitudes could be insightful in finding a more direct way to find the optimal combinations or possibly reduce the size of the exhaustive search.

Last, we acknowledge that the image constraint was extremely tight but the benefit of this was to have a definitive result on which to build before moving on to a weaker constraint. This was accomplished.

LIST OF REFERENCES

- [1] W. B. Pennebaker and J. L. Joan, *JPEG Still Image Data Compression Standard*. New York, NY: Van Nostrand Reinhold, 1993.
- [2] J. Miano, *Compressed Image File Formats, JPEG, PNG, GIF, XBM, BMP*. New York, NY: Association for Computing Machinery Press, 1999.
- [3] K. R. Rao and P. Yip, *Discrete Cosine Transform, Algorithms, Advantages, Applications*. Boston, MA: Academic Press, 1990.
- [4] A. Tescher and R. V. Cox, "An adaptive transform coding algorithm," *Picture Coding Symposim, Asiloma, California*, vol. 47, pp. 20–25, Jan. 1976.
- [5] K. N. Ngan, "Image display techniques using the cosine transform," *IEEE Trans, Acoust, Speech, and Singal Processing*, vol. ASSP-32, NO. 1, pp. 173–177, Feb. 1984.
- [6] Purdue University. "Test images." Dec. 1996. [Online]. Available: http://www.math.purdue.edu/~lucier/test_images/USC

BIBLIOGRAPHY

- Baraiuk, R., "Compressive sensing," *IEEE Signal Processing Magazine*, pp. 118–124, July 2007.
- L. P. Yaroslavsky, G. Shabat, B. G. S. I. A. I. and Fishbain, B., "Nonuniform sampling, image recovery from sparse data and the discrete sampling theorem," *Optical Society of America A*, vol. 26, Issue 3, pp. 566–575, 2009.
- Miano, J., *Compressed Image File Formats, JPEG, PNG, GIF, XBM, BMP*. New York, NY: Association for Computing Machinery Press, 1999.
- N. Ahmed, T. N. and Rao, K. R., "Discrete cosine transform," *IEEE Transactions on Computers*, vol. 23 Issue 1, pp. 90–93, 1974.
- Ngan, K. N., "Image display techniques using the cosine transform," *IEEE Trans, Acoust, Speech, and Singal Processing*, vol. ASSP-32, NO. 1, pp. 173–177, Feb. 1984.
- Pennebaker, W. B. and Joan, J. L., *JPEG Still Image Data Compression Standard*. New York, NY: Van Nostrand Reinhold, 1993.
- Purdue University. "Test images." Dec. 1996. [Online]. Available: http://www.math.purdue.edu/~lucier/test_images/USC
- Rao, K. R. and Yip, P., *Discrete Cosine Transform, Algorithms, Advantages, Applications*. Boston, MA: Academic Press, 1990.
- Tescher, A. and Cox, R. V., "An adaptive transfrom coding algorithm," *Picture Coding Symposim, Asiloma, California*, vol. 47, pp. 20–25, Jan. 1976.