RECONSTRUCTION OF IMAGES OF DIFFERENT FOCI USING FRAMES

A thesis submitted to the faculty of
San Francisco State University
In partial fulfillment of
The requirements for
The degree

Master of Arts
In
Mathematics

by

Mitchell Schoenbrun

San Francisco, California

May 2008

CERTIFICATION OF APPROVAL

I certify that I have read *Investigation into the Reconstruction of Digital Images using Frames* by Mitchell Schoenbrun, and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirements for the degree: Master of Arts in Mathematics at San Francisco State University.

_____

Shidong Li
Professor of Mathematics

_____

Eric Hayashi
Professor of Mathematics

_____

David Ellis
Professor of Mathematics

RECONSTRUCTION OF IMAGES OF DIFFERENT FOCI USING FRAMES

Mitchell Schoenbrun
San Francisco State University
2008

Using digital images as vectors, frame theory is used to numerically reconstruct an image from two generated images with two different focus distances. Dimension invariance is established allowing calculation of an approximate pseudo inverse in a smaller subspace. Proofs are provided showing that the duals of frame constructed as rotational translates of a finite set of vectors are themselves rotational translates, and that the duals of vectors of compact support may be calculated exactly in a smaller dimension. Calculations confirm the expected results and provide insight into solving practical applications.

I certify that the Abstract is a correct representation of the content of this thesis.

_____ _____

Chair, Thesis Committee Date

ACKNOWLEDGEMENTS

I would like to thank professor Shidong Li for suggesting the topic, for his invaluable assistance in the application of frame theory and for his constant encouragement. I'd also like to thank my wife Elva for her patience, support, and occasional willingness to read my drafts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF IMAGES

LIST OF APPENDICES

CHAPTER 1. Introduction

This paper investigates image representation and reconstruction using frames of translates. One application of this study is in the reconstruction of an image from two or more images of different foci. Digital images are treated as vectors for which the intensity value of each pixel is a coordinate coefficient. The two images of different foci are treated as a projection onto a frame from the original in focus image. The reconstruction of the original image is made possible by finding a dual frame. The calculation of the dual frame would normally be numerically intractable. Part of the main focus of this study is about a dimension invariance property that allows the dual calculation using a subspace. The dimension invariance is shown under the compact support assumption on the frame. The reasonableness of this assumption on practical applications is tested.

CHAPTER 2. Frame Theory

Definition: Given a Hilbert space V, a frame is a countable set of vectors $\{f_k\}_{k\in I} \subseteq V$ with constants $0 < A \leq B$ such that

$$\forall f \in V \quad A\|f\|^2 \leq \sum_{k\in I} \left|\langle f, f_k \rangle^2\right| \leq B\|f\|^2 .$$

The constants A and B are respectively known as the lower and upper frame bounds. The spaces in this paper are all finite dimensional over the field $\mathbb{R}$ so for a frame with m elements, the frame inequality becomes.

$$A\|f\|^2 \leq \sum_{k=1}^{m} \left|\langle f, f_k \rangle\right|^2 \leq B\|f\|^2 .^{1}$$

For any finite set of vectors using the space defined by $V = \text{span}\{f_k\}$, we can choose $B = \sum_{k=1}^{m} \|f_k\|^2$ and the upper frame constraint will automatically be satisfied. Similarly, there always exists an A such that the lower constraint will be satisfied.[2]

Given a frame there are three operators of importance; the analysis operator T

$$T : V \to \mathbb{R}^m, \text{ such that } Tf = \{\langle f, f_k \rangle\}_{k=1}^{m},$$

the pre-frame or synthesis operator, a linear mapping which is the adjoint of T

$$T^* : \mathbb{R}^m \to V, \text{ such that } T^*\{c_k\}_{k=1}^{m} = \sum_{k=1}^{m} c_k f_k,$$

and by composing these two operators we get the frame operator

$$S : V \to V, \text{ such that } Sf = T^{*}Tf = \sum_{k=1}^{m} \langle f, f_k \rangle f_k .^{3}$$

The formula $Sf = \sum_{k=1}^{m} \langle f, f_k \rangle f_k$ should be reminiscent of the decomposition of a vector by an orthogonal basis, however the vectors $\{f_k\}$ may not be orthogonal nor linearly independent. From basic frame theory we know that the frame operator S is invertible and self-adjoint and it can be used to represent any vector as

$$f = \sum_{k=1}^{m} \langle f, S^{-1} f_k \rangle f_k = \sum_{k=1}^{m} \langle f, f_k \rangle S^{-1} f_k .^{4}$$

In general, given a frame $\{f_k\}$, there are dual frames $\{g_k\}$ such that

$$\forall f \in V, \quad f = \sum_{k=1}^{m} \langle f, g_k \rangle f_k = \sum_{k=1}^{m} \langle f, f_k \rangle g_k .^{5}$$

The dual where $g_k = S^{-1} f_k$ is called the standard dual.

CHAPTER 3. 2D Frames of Translates in Image Representation

This paper will be dealing with frames in a finite 2D space where the frame elements are integer rotational translates of a finite set of vectors, specifically two prototype vectors. Frames of translates arise naturally when dealing with images and their representation. Let the coordinates of the vector $f_{0,0}$ associated with the origin be $f_{0,0}[x,y]$ so that the translates of $f_{0,0}$ have coordinates

$$f_{i,j}[x,y] = f_{0,0}[x+i, y+j],\ i,j \in \mathbb{Z}.$$

The indexes i and j of $f_{I,j}$ indicate the amount of offset in the x and y direction of the vector relative to $f_{0,0}$. For computational convenience, a "rotational translate" is defined here so that periodicity is assured, ie. for an image of height n and width m, $f[1,y] = f[n+1,y]$, and $f[x,1] = f[x, m+1]$. Equivalently

$$f_{i,j} = \mathcal{R}^{i,j} f_{0,0}.$$

where $\mathcal{R}^{i,j}$ is a 2 dimensional rotational operator.

Taking an image f and projecting it onto a subspace defined by the two vectors $f_{0,0}$ and $f'_{0,0}$, and their translates $f_{i,j}$ and $f'_{i,j}$, the result is a vector in a subspace which is the span of the frame $\{f_{i,j}, f'_{i,j}\}$. This projected vector may be written as $(\langle f, f_{0,0}\rangle, \langle f, f_{0,1}\rangle, \ldots, \langle f, f_{n,m}\rangle, \langle f, f'_{0,0}\rangle, \langle f, f'_{0,1}\rangle, \ldots, \langle f, f'_{n,m}\rangle)$, which contains the coefficients of two images, each at a different focus.

The original image can then be reconstructed using the formula.

$$f = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left\langle f, f_{i,j}\right\rangle g_{i,j} + \sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left\langle f, f'_{i,j}\right\rangle g'_{i,j}$$

where the frame $\{g_{i,j}, g'_{i,j}\}$ is a dual of $\{f_{i,j}, f'_{i,j}\}$. Because the original vectors are all translates, the standard dual vectors will themselves all be translates. A proof of this follows. First it is shown that $\mathcal{R}^{i,j}$ and S commute.

Lemma 3.1: The operators $\mathcal{R}^{i,j}$ and S commute.

Proof: Recall that every frame $\{h_k\}_{k\in I}$ has a standard dual frame $\{S^{-1}h_k\}_{k\in I}$.

So for each k, $g^k_{0,0} = S^{-1}f^k_{0,0}$ is the dual of $f^k_{0,0}$.

By our definition of the frame operator,

$$Sf^{\ell}_{0,0} = \sum_k\sum_i\sum_j\left\langle f^{\ell}_{0,0}, f^k_{i,j}\right\rangle f^k_{i,j}.$$

Applying $\mathcal{R}^{i,j}$ to both sides of this equation we find that

$$\mathcal{R}^{p,q}Sf^{\ell}_{0,0} = \mathcal{R}^{p,q}\sum_k\sum_i\sum_j\left\langle f^{\ell}, f^k_{i,j}\right\rangle f^k_{i,j} =$$

$$\sum_k\sum_i\sum_j\left\langle f^{\ell}_{0,0}, f^k_{i,j}\right\rangle \mathcal{R}^{p,q}f^k_{i,j} = \sum_{k=1}^{q}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left\langle f^{\ell}_{0,0}, f^k_{i,j}\right\rangle f^k_{i+p,j+q}.$$

Now note that $\forall p,q\in I,\ \left\langle f,g\right\rangle = \left\langle \mathcal{R}^{p,q}f, \mathcal{R}^{p,q}g\right\rangle$ so we have

$$\sum_k\sum_i\sum_j\left\langle f^{\ell}_{0,0}, f^k_{i,j}\right\rangle f^k_{i+p,j+q} = \sum_k\sum_i\sum_j\left\langle \mathcal{R}^{p,q}f^{\ell}_{0,0}, \mathcal{R}^{p,q}f^k_{i,j}\right\rangle f^k_{i+p,j+q} =$$

$$\sum_k\sum_i\sum_j\left\langle f^{\ell}_{p,q}, f^k_{i+p,j+q}\right\rangle f^k_{i+p,j+q} = \sum_k\sum_i\sum_j\left\langle f^{\ell}_{p,q}, f^k_{i,j}\right\rangle f^k_{i,j} = S\mathcal{R}^{p,q}f^{\ell}_{0,0}. \quad \square$$

So $\mathcal{R}^{p,q} S f^{\ell}_{0,0} = S\mathcal{R}^{p,q} f^{\ell}_{0,0}$ and therefore the S and $\mathcal{R}^{i,j}$ operators commute.

Theorem 3.2: Given $\{\mathcal{R}^{i,j} f^{k}_{0,0}\} = \{f^{k}_{i,j}\}$, a frame sequence of integer rotational translates of a finite set of q vectors $f^{k}_{0,0}$, there are vectors $g^{k}_{0,0}$ whose integer rotational translates $\{\mathcal{R}^{i,j} g^{k}_{0,0}\} = \{g^{k}_{i,j}\}$ are a dual frame sequence.

Proof: Since S is bijective, S^{-1} and $\mathcal{R}^{i,j}$ commute, it follows that

$$S^{-1} f^{k}_{i,j} = S^{-1}\mathcal{R}^{p,q} f^{k} = \mathcal{R}^{p,q} S^{-1} f^{k} = \mathcal{R}^{p,q} g^{k}. \qquad \square$$

This result will be extremely helpful computationally, as it means that by finding the duals of the prototype elements f^{k}, the entire dual frame can be generated through translates.

CHAPTER 4.

Numerical Complexity of Dual Frame Evaluation and an Efficient Algorithm

4.1 The Pseudo-Inverse and its Computation

Given an m × n image, the vectors in our space will have *mn* coefficients, and the matrix representation of *T* will be an *mn* × 2*mn* matrix, with each row having the coefficients of a frame member. Given this mapping from $E:\mathbb{R}^{mn}\rightarrow\mathbb{R}^{2mn}$ a mapping $E^{\dagger}:\mathbb{R}^{2mn}\rightarrow\mathbb{R}^{mn}$ with the properties of a dual frame needs to be calculated. Note that *E* is not surjective. Since it is injective it has an inverse E^{-1} which can be extended to the mapping $E^{\dagger}$ by setting $E^{\dagger}(y+z) = E^{-1}(y)$ if $y \in \text{Range}(E)$ and $z \in \text{Range}(E)^{\perp}$. $E^{\dagger}$ is known as the Moore-Penrose pseudo-inverse.[6] Calculation of this inverse requires the singular value decomposition of a matrix, which has complexity $O(n^3)$. A more complete description of this calculation is included in Appendix A.

4.2 Computation of Duals with Compact Support - Dimension Invariance

It will become evident later in this paper that the spaces whose duals must be computed have a dimension equal to the number of pixels in the digital images. The majority of images tested in this paper are a rather small 384 x 512 pixels, for a dimension of approximately 200,000. The pseudo-inverse that will need to be computed will therefore require the inversion of an already 200,000 x 400,000 entry matrix. The computer

memory requirements just to store this matrix are about 640 gigabytes. This would of course be quite unwieldy, and the time to perform the calculation would be astronomical. Fortunately this computation can be vastly simplified. For vectors with compact support, given a space with large enough dimension, the coefficients of their respective dual vectors should not vary with the dimension of the space. This is called dimension invariance, and a proof follows.

Definition 4.3: Define the minimum covering ball $MCB(f)$ of a function f as follows. Let $\overline{B}_r$ be a closed ball with radius r such that $(x,y) \in \overline{B}_r$ implies $f(x,y) = 0$ and for which $r \leq q$ for all closed balls $\overline{B}_q$ that satisfy the same condition. Let $MCB(f) = \overline{B}_r$ be the minimum covering ball of f and let $diam\text{-}MCB(f) = 2r$ be its diameter. If a finite r exists then the function has compact support with $supp(f) \subseteq MCB(f)$.

Definition 4.4: Define an *M×N-2D* space as a space of dimension *MN* where the coordinates are indexed as an *M×N* matrix.

Definition 4.5: Define an *extension* mapping as a mapping from a vector in an *N×M-2D* space to an $(N+\alpha)\times(M+\beta)$*-2D* space with $\alpha \geq 1$ and $\beta \geq 1$ and where the vector's support does not span the matrix boundaries. The mapping takes each coordinate in the *N×M*

space to its respective coordinate in the $(N+\alpha)\times(M+\beta)$ space and each new coordinate in the $(N+\alpha)\times(M+\beta)$ space is set to zero.

Theorem 4.6: Given a frame in a finite $N\times N$-*2D* space with frame members that are rotational integer translates of two unique generating vectors, let the sum of the diameters of the minimum covering balls of each generating vector and its dual be less than N, $diam\text{-}MCB(f) + diam\text{-}MCB(g) \le N$. If the generating vectors are chosen so that their support does not overlap the boundaries of the matrix then an *extension* mapping of these vectors to a larger *2D* space will preserve duals. That is, if the vectors and dual vectors have compact support in a large enough space then dimension invariance will be achieved for any larger space.

Proof: Let $\{\mathcal{R}^{ij}f^1, \mathcal{R}^{ij}f^2\} = \{f_{ij}{}^1, f_{ij}{}^2\}$ $i,j \in \{0,\ldots,N\text{-}1\}$ be a set of frame vectors in the $N\times N$-*2D* space V spanned by this frame with N odd. Let f^1 and f^2 be the corresponding members of $\{f_{ij}{}^1, f_{ij}{}^2\}$ for which $MCB(f^1)$ *and* $MCB(f^2)$ are centered in the space. Centered means that each coordinate of the centers of $MCB(f^1)$ and $MCB(f^2)$ are in the interval $[(N\text{-}1)/2\text{-}½,(N\text{-}1)/2+½]$, ensuring that the support does not overlap the *2D* matrix boundaries. Let $\{\mathcal{R}^{ij}g^1, \mathcal{R}^{ij}g^2\} = \{g^1{}_{ij}, g^2{}_{ij}\}$ be the standard dual vectors for the frame $\{f_{ij}{}^1, f_{ij}{}^2\}$ in V, chosen so that $MCB(g^1)$ and $MCB(g^2)$ are also centered.

Let the vectors $f^1,\ f^2,\ g^1,\ g^2$ and their translates have compact support with

$$diam\text{-}MCB\left(f^k\right) + diam\text{-}MCB\left(g^\ell\right) \le N^*.$$

Let f^{1+} be a vector in a $P\times Q$-$2D$ space V^+ with $N < P,Q$, $f^{1+}(x,y) = f^1(x,y)$ for $0 \le x,y < N$, $f^{1+}(x,y) = 0$ for $x,y \ge N$ and $V^+ = span\{f_{ij}^{\ 1+},\ f_{ij}^{\ 2+}\}$ $i \in \{0,\ldots,P\text{-}1\}$, $j \in \{0,\ldots,Q\text{-}1\}$. That is $f^k \rightarrow f^{1+}$ is an *extension* mapping from V to V^+. Define f^{2+}, g^{1+} and g^{2+} in the corresponding manner.

The frame decomposition equation gives us the following for all f.

$$f = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(\left\langle f, g_{ij}^1\right\rangle f_{ij}^1 + \left\langle f, g_{ij}^2\right\rangle f_{ij}^2\right)$$

Let $c = (N\text{-}1)/2$ and substitute $f^1{}_{cc}$ for f.

$$f_{cc}^1 = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(\left\langle f_{cc}^1, g_{ij}^1\right\rangle f_{ij}^1 + \left\langle f_{cc}^1, g_{ij}^2\right\rangle f_{ij}^2\right)$$

By the construction of f^{1+}, f^{2+}, g^{1+} and g^{2+}

$$f_{cc}^{1+} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(\left\langle f_{cc}^{1+}, g_{ij}^{1+}\right\rangle f_{ij}^{1+} + \left\langle f_{cc}^{1+}, g_{ij}^{2+}\right\rangle f_{ij}^{2+}\right)$$

By the compact support requirement, for $N \le k < P$ or $N \le l < Q$

$$\left\langle f_{cc}^{1+}, g_{kl}^{1+}\right\rangle = \left\langle f_{cc}^{1+}, g_{kl}^{2+}\right\rangle = 0$$

This implies that

$$f_{cc}^{1+} = \sum_{i=0}^{P-1}\sum_{j=0}^{Q-1}\left(\left\langle f_{cc}^{1+}, g_{ij}^{1+}\right\rangle f_{ij}^{1+} + \left\langle f_{cc}^{1+}, g_{ij}^{2+}\right\rangle f_{ij}^{2+}\right)$$

By a translation of the coordinates, $x \rightarrow x + p$ and $y \rightarrow y + p$,

$$f^{1+}_{c+p,c+q} = \sum_{i=p}^{P-1+p} \sum_{j=q}^{Q-1+q} \left(\left\langle f^{1+}_{c+p,c+q}, g^{1+}_{i+p,\,j+q} \right\rangle f^{1+}_{ij} + \left\langle f^{1+}_{c+p,c+q}, g^{2+}_{i+q,\,j+q} \right\rangle f^{2+}_{i+p,\,j+q} \right)$$

Then by a change of variables in the summations, $i \rightarrow i - p$ and $j \rightarrow j - q$,

$$f^{1+}_{c+p,c+q} = \sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} \left(\left\langle f^{1+}_{c+p,c+q}, g^{1+}_{ij} \right\rangle f^{1+}_{ij} + \left\langle f^{1+}_{c+p,c+q}, g^{2+}_{ij} \right\rangle f^{2+}_{ij} \right)$$

Since $f^{1+}_{c+p,c+q} = \mathcal{R}^{pq} f_{cc}{}^{1+}$, the extended duals are the duals of the extended frame members.

This equation is true for all $\mathcal{R}^{pq} f_{cc}{}^{1+}$ and is therefore true for all the translates of f^{1+}.

The same steps work equally well for f^{2+} so it is true for all frame members, and therefore by linearity it is true for their span V^{+}. So the frame $\{f_{ij}{}^{1+}, f_{ij}{}^{2+}\}$ will have $\{g_{ij}{}^{1+}, g_{ij}{}^{2+}\}$ as its standard dual. This conclusion will hold for a slightly more general setting as in the following theorem.

Corollary 4.7: Theorem 4.3 holds for $\{f_{ij}{}^{k}\}$ where $k \in \{1, 2, \ldots, n\}$.

Proof: Apply the same steps in 4.3 to $f = \sum_{k=1}^{n} \sum_{i=1}^{N} \sum_{j=1}^{N} \left\langle f, g^{k}_{ij} \right\rangle f^{k}_{ij}$.

---

* Normally the vectors will be centered exactly and $< N$ will be sufficient.

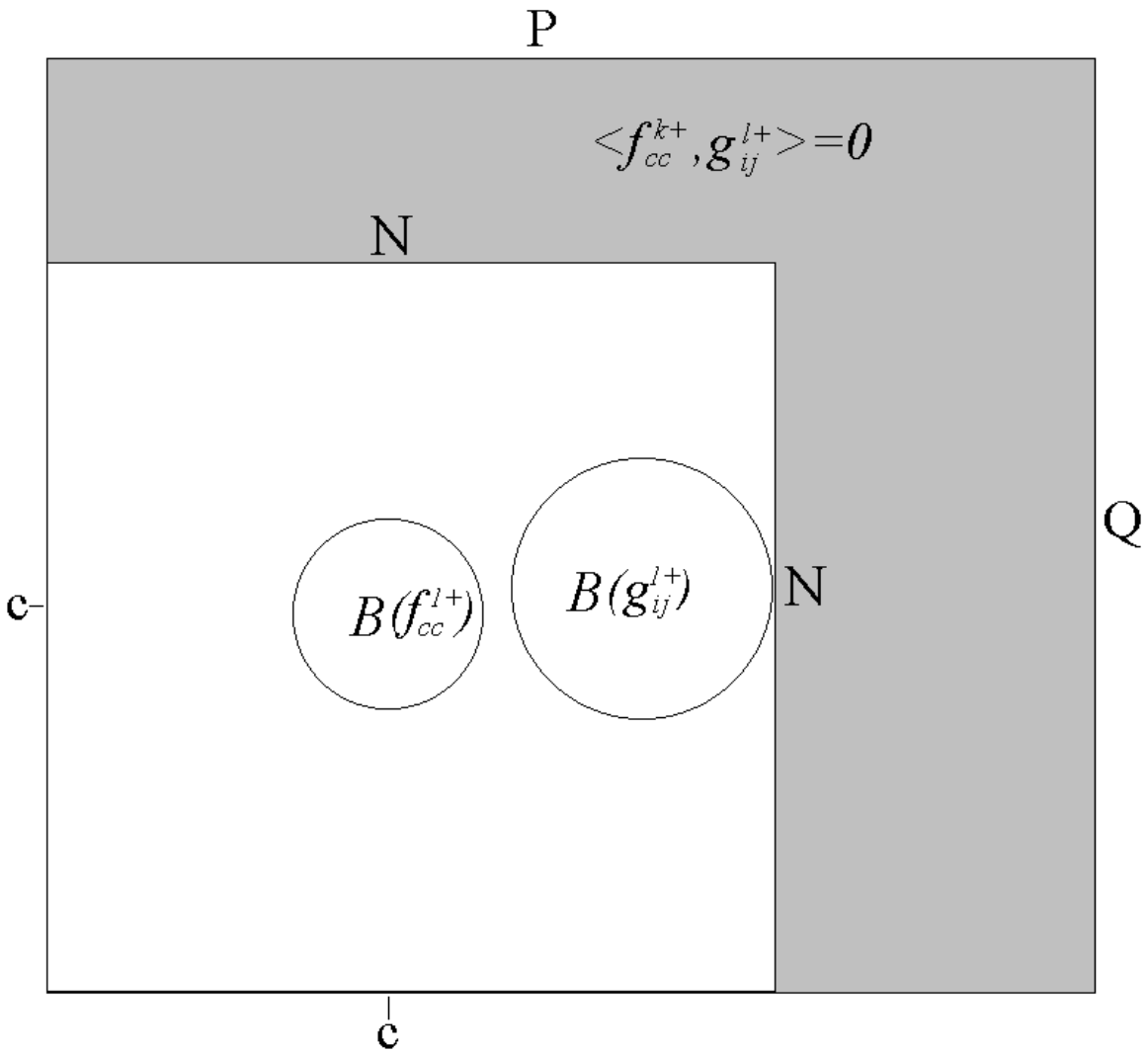

Figure 1. Illustration of Theorem 4.3

CHAPTER 5.

Implementation and Application of Image Combinations of Two different Foci

5.1 Digital Images as Vectors

In this paper, digital black and white images are represented as m×n matrices $[a_{i,j}]$ $0 \le i < m$, $0 \le j < n$, which can also be viewed as *mn*-tuples, or vectors of dimension *mn*.

$$(a_{1,1}, \ldots, a_{n,1}, a_{2,1}, \ldots, a_{2,n}, \ldots, a_{m,1}, \ldots, a_{m,n})$$

In most practical applications, the elements of these matrices are integers $a_{i,j} \in \{0,1,\ldots,255\}$, however the intermediate computations treat them as reals, $a_{i,j} \in [0,255]$. To simplify some calculations, the 2-dimensional image wrapped around with $a_{i,j} = a_{i+m,j}$, and $a_{i,j} = a_{i,j+n}$. The justification is that where unknown image data that is outside of the boundary of the image is needed for computations, it is reasonable to use known data that is part of the image. The inner product applied to this vector space is the 2-dimensional dot product

$$\langle a, b \rangle = \sum_{k=0}^{m} \sum_{l=0}^{n} a_{k,l} b_{k,l}.$$

Viewed as *mn*-tuples, this is the standard dot product.

5.2 Frame Vectors

The two prototype vectors for the frame are generated as follows

$$f_{0,0}^{k}(i,j) = \left[Ce^{\frac{-(i^2+j^2)}{\sigma_k^2}} \right]_{i=-\frac{m-1}{2}, j=-\frac{n-1}{2}}^{\frac{m-1}{2}, \frac{n-1}{2}} \qquad k = 1,2$$

The x,y indices are over the integers [0, …, m-1] and [0, …, n-1], so there are $2mn$ vectors in the frame. The C constant is a normalizing constant. Each of these matrixes is converted to an mn-tuple or row vector. Using these row vectors, an $mn \times 2mn$ transformation matrix is constructed. For each pixel either the appropriate "in focus" or "out of focus vector" is used depending on the value of the mapping function. When this transformation matrix operates on the mn column vector representing the image matrix, a $2mn$ column vector is computed. Each half of this vector represents an image focused at one of the two distances. In a real world application this vector contains the two images obtained by a camera, one focused at the foreground distance and one focused at the background distance.

This matrix is the S operator which is invertible. By computing S^{-1} the original in focus image may be reconstructed by operating on these vectors. The following figure shows three-dimensional graphs of a typical vector used in this paper.

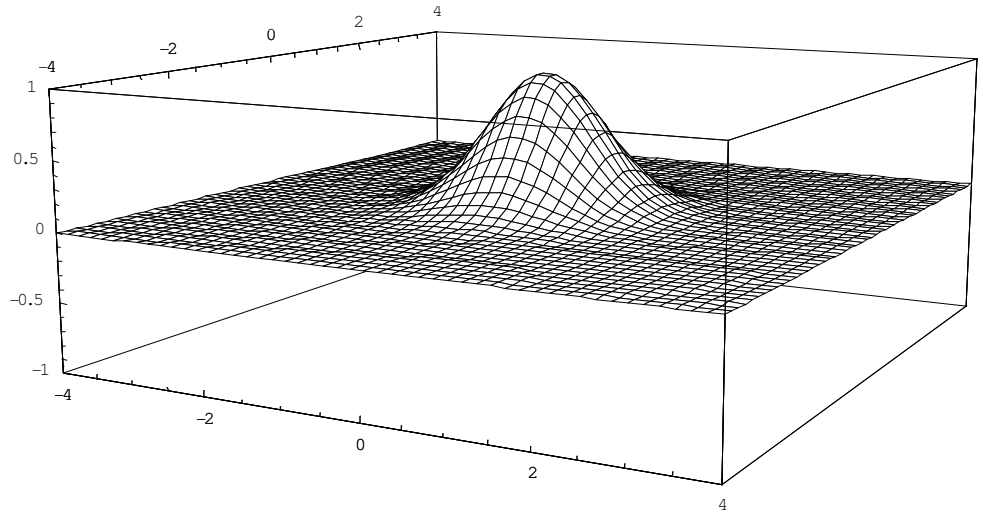

Figure 2. A Gaussian

5.2 Creation of the Testing images

A picture was chosen for use in simulation studies. A focus map was created using a GUI program that displayed the image as a background. The focus map consisted of a single 0 or 1 for each pixel in the original image indicating whether the corresponding pixel was to be considered in the foreground or background focus plane. Two normalized, digitized $n \times n$ Gaussian vectors, f^1 and f^2 were prepared. The testing images were created by projecting the original image onto the frame systems formed by the rotational translations of these two vectors. Nineteen trials were performed using different pairs of vectors. The first vector was always created with $\sigma = .15$. The second vector was varied in each trial with the Gaussian sigma constant set as $\sigma = .15 + .18k$,

with k varying from 1 to 19. Using a larger value of k caused a flatter Gaussian and the resulting images were more out of focus. The conversion of each pixel in the images was calculated as the dot product of one of the two vectors offset into the original image. This simulated a digital camera photographing two images at two different focus settings. The focus map was used to decide which of the vectors the translated vectors f^1 or f^2 were used in the dot product, so it determined which regions of the original image were in or out of focus.

5.4 Computation Issues

The required computation described so far consists of finding the pseudo-inverse of an $mn \times 2mn$ matrix. For an image with 384×512 pixels, this would involve finding the pseudo-inverse of an approximately $200{,}000 \times 400{,}000$ matrix, one which has about 80,000,000,000 elements. The memory requirements just to hold this matrix are about 640 gigabyte, which is beyond the capacity of most modern PC's. The time to calculate the pseudo-inverse of this matrix on a modern PC would probably need to be measured in millennia.

If the frame vectors happens to have compact support, then the dual vectors could be calculated using a much smaller matrix. Unfortunately the actual function of a real camera does not have this property, nor would the duals. However the tails of both of the original function, and the dual vectors should decrease exponentially. Therefore they can

be expected to act approximately like compactly supported functions if the assumed support is large enough. The following two figures show a dual calculated as part of this investigation.

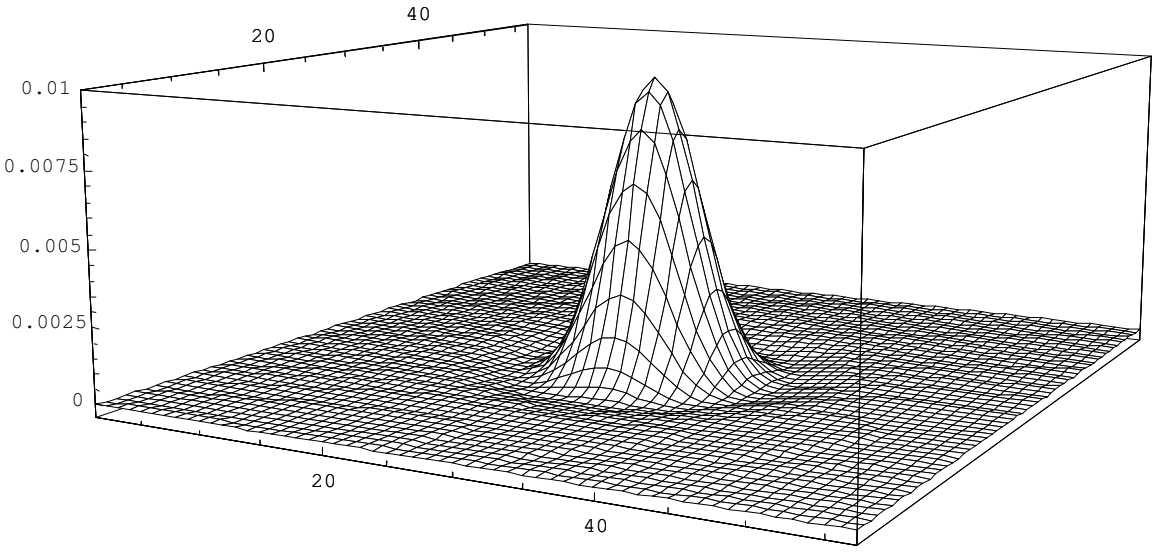

Figure 3. A Dual, the Entire Function

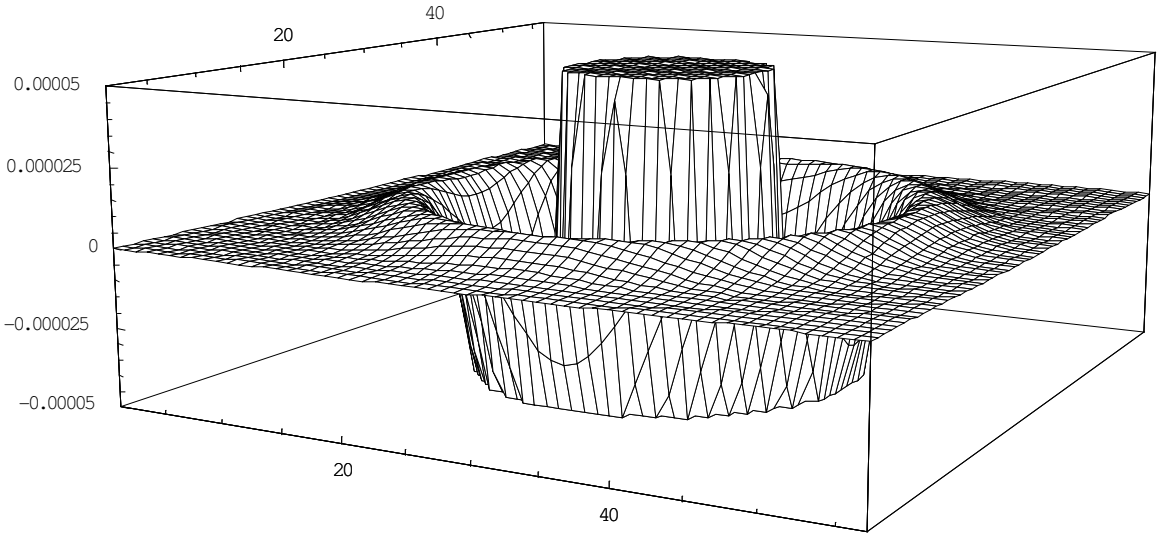

Figure 4. A Close Up View of the Dual

There are two levels of accuracy of interest with respect to when the vectors approximate compact support. One is the numerical accuracy that the IEEE-754 floating-point values used in all calculations are capable of providing. The IEEE format has 64 bits allocated to the mantissa of the value that translates to about 16 decimal places. The value of each pixel ranges from 0 to 255, so accuracy's better than 10^{-13} are not expected except by coincidence. So what size matrix is needed to obtain this accuracy?

A more practical question is what size matrix is needed to accurately reproduce the original photograph? This calculation will only require a accuracy of about 10^{-3} assuming that an average error of less than .1, will produce an image that is indistinguishable from and in most cases identical to the original. Even less accuracy may be visually acceptable.

5.5 Computing the Focus Map

When simulating the affect of taking a picture by projection, a determination must be made for each pixel as to whether it will use the in-focus or out-of-focus frame members. This was done using a focus map that had a binary value for each pixel indicating whether the pixel was in or out of focus. This is provided a priori for this study as part of the simulation, but in a real world application this would not be the case. It would however be possible to generate this map. One solution for generating the focus map comes from known image processing algorithms. To find object edges in the original

two images, the following procedure can be applied. The images can be smoothed using a two dimensional Gaussian, followed by finding the 2nd derivative. Zero crossings of the result indicate object edges. This calculation is simplified by smoothing with the 2nd derivative of the Gaussian.[7] Edges will divide the image into separate regions. Comparing the regions from each separate image, the values within an out of focus region will be more uniform than those of an in-focus region.

CHAPTER 6. Results

6.1 First Test

In an attempt to confirm the theory, a very small image of size 32×32 pixels was used. The original image is too small to view easily, so that each pixel in the image below has been enlarged for clarity.

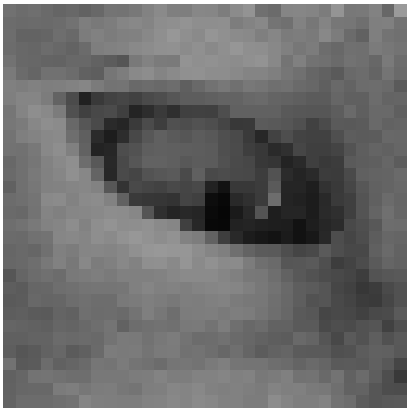

Image 1. Original Cat's Eye

This image was used to generate two partially out of focus images using a very simple focus map that defocuses all pixels to the right or to the left of the center. Defocusing was done by projecting the original image onto a 32×32 dimension space, whose frame vectors were constructed from a Gaussian. The σ value of the Gaussian chosen from the formulae $\sigma = .15 + (.18)\,k$, using an integer $k = 1, \ldots, 19$. In this first test I used k=19 or σ=2.57.

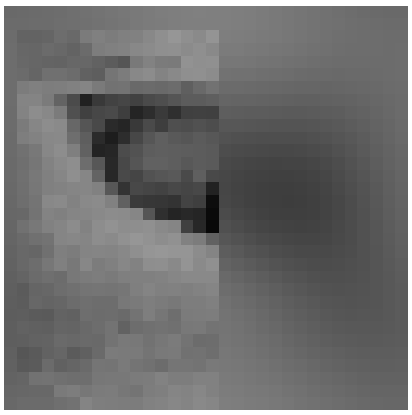

Image 2. Left in Focus

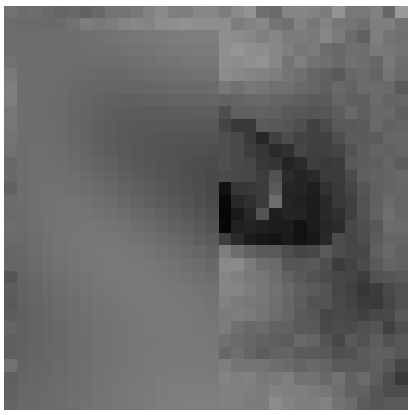

Image 3. Right in Focus

Note that the reconstructed image below is indistinguishable from the original.

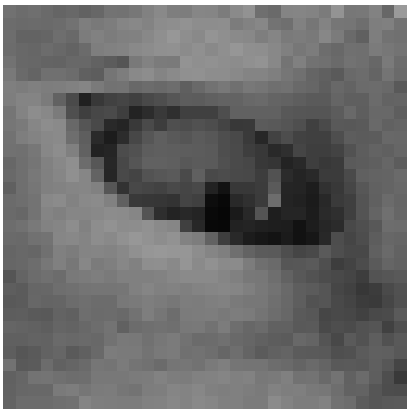

Image 4. Reconstructed Image

In the test displayed here the values $\sigma = 3.57$ (k=19) was used. The space that the duals were calculated in had dimension 32×32. The maximum pixel error in this trial was 0.256, and the average pixel error was 0.074. The quantized (integer) pixel values for the reconstructed image were identical to the original image. This was not surprising and it confirmed the correctness of the software as well as the use of frame theory. Next a more realistic test was run using a more realistically sized image.

6.2 Reconstruction

The 384×512 pixel image shown below was chosen for all trials.

Image 5. Original image

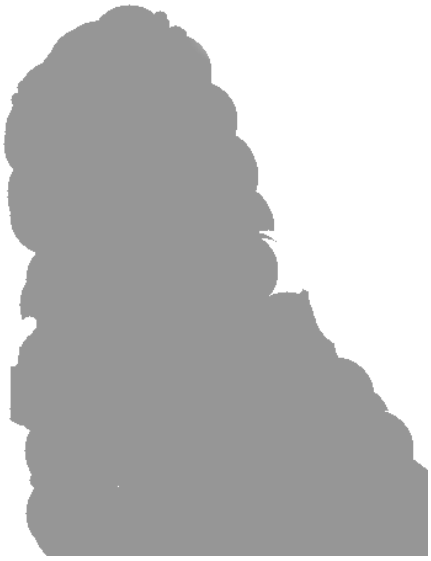

Image 6. Focus Map (Representation)

A program read in this image and also the focus map. Image 6 is a visual representation of the focus map used to show which regions of the image were simulated to be in the foreground and background. Two different unfocused images were created using this map, one focused in the foreground and one focused in the background. Dual vectors were computed in subspaces of each of the 19 trials described in section 5.2. For each frame the duals were computed in subspaces ranging from dimension 17×17 through 57×57 in steps of 2. Computing the duals in small dimension subspace was fast, but in large dimension it because very time consuming, up to 2 or 3 days. After computing the duals a representative pair, g_1 and g_2, of the dual vectors were identified and saved separately. All other duals were translates of these two vectors as shown by theorem 3.1.

These duals were saved on the computer's disk so that the reconstruction tests could be repeated without the long computation each time. The dual vectors were logically zero filled to match the dimension of the image. Knowledge of which coordinates were zero was used to optimize the final step in reconstruction. Translations of the representative duals were then used to reconstruct the image from the two unfocused images. Finally the reconstructed image data was compared to the original data. An error value was calculated for each trial as the average of the absolute value of the differences of the original and reconstructed pixels. The reconstruction program was used repeatedly for each frame-subspace pair. The purpose of all these trial was to find the minimum subspace for which each frame showed dimension invariant behavior. This average error for each trial is listed in table 1 below.

Image 7. Foreground in focus k=19

Image 8. Background in focus k=19

Image 9. Reconstructed image using support width 23

6.3 Evaluation of Reconstruction Computations

The following table shows the average error to one decimal place for each combination of σ (frame pair) and each subspace dimension. The first thing to notice is the boldface region of the Table 1 that appears in part 1. For *k* less than 7, at some dimension the values stop decreasing below 10^{-12}. Given the range of values for pixels of 0 to 255, these values near 10^{-12} are indistinguishable from zero. This minimum error is the round off error found with the IEEE-754 floating-point variables used in the calculations. To reconstruct the image to better accuracy, higher precision variables would be needed. This leveling off of the errors indicates where dimension invariance is achieved. The second region to notice in Table 1, part 2 is the italicized region that is found in the upper right hand corner of the table. These errors represent reconstruction that would be less

than perfect with respect to the quantized integer values of pixels found in most digital images. Even in this region the errors in reconstruction will be hard to notice visually. This is an important finding for practical reasons. The 57×57 dual calculation represents the practical limits of computation with the computer hardware used. The computation of duals with this size subspace took 2-3 days and maximum accuracy was not reached for any $\sigma > 1.08$ ($k>6$). On the other hand, the 57×57 dual calculation took less than a minute and would clearly be acceptable for most practical purposes. The boundaries between these regions are somewhat arbitrary, however the general shape is clear, and extrapolation to larger subspaces can be calculated.

k	1	2	3	4	5	6	7	8	9	10	11	12
Support n×n												
17	**5.8E-13**	2.0E-09	5.3E-07	1.2E-05	9.4E-05	7.4E-04	2.5E-03	2.2E-03	7.0E-03	2.3E-02	4.6E-02	7.0E-02
19	**1.3E-13**	2.8E-10	1.4E-07	3.8E-06	5.5E-05	6.4E-05	1.1E-03	2.7E-03	2.4E-03	5.6E-03	1.9E-02	3.8E-02
21	**3.9E-13**	1.4E-11	4.6E-09	8.7E-07	5.3E-06	9.8E-05	1.6E-04	1.4E-03	2.9E-03	2.5E-03	4.6E-03	1.6E-02
23	**1.8E-13**	**1.4E-12**	4.0E-09	1.6E-07	4.1E-06	3.5E-05	1.1E-04	3.6E-04	1.7E-03	3.1E-03	2.7E-03	4.0E-03
25	**2.3E-12**	**6.0E-13**	2.7E-10	5.8E-08	1.3E-06	3.0E-06	7.9E-05	9.2E-05	5.9E-04	2.0E-03	3.3E-03	2.9E-03
27	**1.7E-12**	**1.1E-12**	9.1E-11	5.8E-09	1.2E-07	4.4E-06	2.3E-05	1.2E-04	9.4E-05	8.5E-04	2.3E-03	3.4E-03
29	**2.7E-12**	**1.7E-12**	1.4E-11	3.5E-09	1.4E-07	1.7E-06	3.2E-06	5.9E-05	1.3E-04	1.9E-04	1.1E-03	2.5E-03
31	**1.3E-12**	**7.6E-13**	**2.0E-12**	1.7E-10	2.4E-08	1.7E-07	4.6E-06	1.5E-05	9.8E-05	1.2E-04	3.5E-04	1.4E-03
33	**2.8E-12**	**8.3E-13**	**1.0E-12**	2.0E-10	7.6E-09	1.9E-07	2.1E-06	4.0E-06	4.3E-05	1.3E-04	9.6E-05	5.3E-04
35	**3.1E-12**	**1.3E-12**	**2.1E-12**	1.2E-11	3.7E-09	8.2E-08	3.7E-07	4.8E-06	9.8E-06	7.8E-05	1.4E-04	1.2E-04
37	**2.7E-12**	**2.5E-12**	**1.8E-12**	1.1E-11	2.9E-10	9.5E-09	2.0E-07	2.5E-06	4.7E-06	3.2E-05	1.1E-04	1.3E-04
39	**3.1E-12**	**7.0E-13**	**2.0E-12**	**2.4E-12**	3.0E-10	8.3E-09	1.5E-07	6.2E-07	5.1E-06	6.7E-06	6.2E-05	1.4E-04
41	**3.3E-12**	**2.1E-12**	**2.6E-12**	**2.7E-12**	8.4E-11	4.0E-09	4.7E-08	1.7E-07	2.8E-06	5.3E-06	2.4E-05	9.6E-05
43	**3.8E-12**	**3.6E-12**	**3.0E-12**	**8.5E-13**	1.1E-11	5.3E-10	5.8E-09	2.1E-07	9.0E-07	5.3E-06	5.0E-06	4.9E-05
45	**3.7E-12**	**4.5E-12**	**3.7E-12**	**2.1E-12**	9.7E-12	3.6E-10	8.6E-09	1.0E-07	1.6E-07	3.1E-06	5.7E-06	1.7E-05
47	**4.2E-12**	**4.4E-12**	**3.8E-12**	**2.6E-12**	**2.7E-12**	1.9E-10	4.2E-09	2.6E-08	2.3E-07	1.2E-06	5.4E-06	4.4E-06
49	**3.3E-12**	**7.1E-12**	**4.8E-12**	**1.6E-12**	**3.3E-12**	2.9E-11	8.0E-10	7.5E-09	1.6E-07	2.3E-07	3.4E-06	6.1E-06
51	**4.4E-12**	**4.7E-12**	**3.1E-12**	**2.1E-12**	**3.1E-12**	1.5E-11	3.6E-10	8.7E-09	6.8E-08	2.3E-07	1.5E-06	5.6E-06
53	**6.0E-12**	**5.4E-12**	**4.3E-12**	**2.2E-12**	**2.1E-12**	8.8E-12	2.9E-10	4.4E-09	1.4E-08	3.8E-07	2.1E-07	3.6E-06
55	**2.9E-12**	**3.2E-12**	**3.2E-12**	**3.1E-12**	**2.8E-12**	**1.8E-12**	9.6E-11	1.1E-09	9.1E-09	1.2E-07	2.5E-07	1.7E-06
57	**5.8E-12**	**7.0E-12**	**5.7E-12**	**1.7E-12**	**2.3E-12**	**1.0E-12**	1.1E-11	3.2E-10	8.7E-09	4.3E-08	2.5E-07	5.5E-07

Table 1. Part 1. Error Values for Each Support-*k* Pair (continued on the next page)

k	13	14	15	16	17	18	19
Support *n*×*n*							
17	9.2E-02	*1.1E-01*	*1.2E-01*	*1.2E-01*	*1.2E-01*	*1.1E-01*	*1.0E-01*
19	6.1E-02	8.2E-02	*1.0E-01*	*1.1E-01*	*1.2E-01*	*1.2E-01*	*1.2E-01*
21	3.3E-02	5.3E-02	7.4E-02	9.3E-02	*1.1E-01*	*1.2E-01*	*1.2E-01*
23	1.4E-02	2.8E-02	4.7E-02	6.7E-02	8.6E-02	*1.0E-01*	*1.1E-01*
25	3.5E-03	1.2E-02	2.5E-02	4.2E-02	6.0E-02	7.9E-02	9.7E-02
27	3.1E-03	3.2E-03	1.0E-02	2.2E-02	3.7E-02	5.5E-02	7.3E-02
29	3.6E-03	3.2E-03	3.0E-03	8.9E-03	1.9E-02	3.3E-02	5.0E-02
31	2.8E-03	3.7E-03	3.4E-03	2.8E-03	7.8E-03	1.7E-02	3.0E-02
33	1.6E-03	3.0E-03	3.8E-03	3.5E-03	2.8E-03	7.0E-03	1.6E-02
35	7.3E-04	1.9E-03	3.1E-03	3.9E-03	3.6E-03	2.8E-03	6.2E-03
37	2.2E-04	9.4E-04	2.1E-03	3.3E-03	4.0E-03	3.7E-03	2.9E-03
39	1.1E-04	3.5E-04	1.2E-03	2.3E-03	3.5E-03	4.1E-03	3.8E-03
41	1.5E-04	1.1E-04	5.0E-04	1.4E-03	2.5E-03	3.6E-03	4.2E-03
43	1.3E-04	1.5E-04	1.5E-04	6.7E-04	1.6E-03	2.7E-03	3.8E-03
45	7.9E-05	1.5E-04	1.3E-04	2.3E-04	8.4E-04	1.8E-03	2.9E-03
47	3.9E-05	1.1E-04	1.6E-04	1.2E-04	3.5E-04	1.0E-03	2.0E-03
49	1.3E-05	6.6E-05	1.4E-04	1.6E-04	1.2E-04	4.8E-04	1.2E-03
51	4.4E-06	3.1E-05					
53	6.4E-06						
55							
57	3.8E-06	6.5E-06	7.6E-06	4.4E-05	1.1E-04	1.7E-04	1.6E-04

Table 1. Part 2. Error Values for Each Support-*k* Pair (continued from the previous page)

Table 2 below shows an approximate but meaningful lower estimate for the dimension where dimension invariance is achieved. These values are taken from the wandering line that borders the top of the boldface region of Table 1 part 1. The values show the total support needed but they do not show what the individual supports are for the original vector and its dual.

σ Index	Sigma	Support $n \times n$	
1	0.33	11	*
2	0.51	23	
3	0.69	31	
4	0.87	39	
5	1.05	47	
6	1.23	55	
7	1.41	63	*
8	1.59	71	*
9	1.77	79	*
10	1.95	87	*
11	2.13	95	*
12	2.31	103	*
13	2.49	111	*
14	2.67	119	*
15	2.80	127	*
16	3.03	135	*
17	3.21	143	*
18	3.39	151	*
19	3.57	159	*
* Estimated			

Table 2. Support Values Determined from Table 1

To estimate the separate individual original vector/dual vector supports, the following two tables were constructed. The table entries show the absolute value of the maximum coefficient found in each vector outside of the area of support. Table 3 is for the original vectors and table 4 is for the dual vectors. Once the tables were constructed, values where the magnitude of the two vectors were about equal were chosen such that the total support was equal to or greater than the support required by table 2. This magnitude is illustrated in the following two tables by the wandering line passing through the tables.

k	1	2	3	4	5	6	7	8
Support *n*×*n*								
0	9.6E-01	6.0E-01	3.3E-01	2.1E-01	1.4E-01	1.1E-01	8.0E-02	6.3E-02
2	9.7E-03	8.7E-02	1.2E-01	1.1E-01	9.2E-02	7.6E-02	6.2E-02	5.2E-02
4	1.0E-08	2.7E-04	5.0E-03	1.5E-02	2.4E-02	2.8E-02	2.9E-02	2.9E-02
6	1.1E-18	1.8E-08	2.6E-05	5.5E-04	2.4E-03	5.4E-03	8.3E-03	1.1E-02
8	1.2E-32	2.6E-14	1.7E-08	5.4E-06	1.0E-04	5.3E-04	1.4E-03	2.7E-03
10	1.4E-50	8.0E-22	1.3E-12	1.4E-08	1.7E-06	2.7E-05	1.5E-04	4.5E-04
12	1.6E-72	5.3E-31	1.3E-17	9.9E-12	1.2E-08	7.2E-07	9.4E-06	5.1E-05
14	1.9E-98	7.4E-42	1.5E-23	1.8E-15	3.2E-11	9.8E-09	3.6E-07	3.9E-06
16	2.3E-128	2.2E-54	2.2E-30	9.2E-20	3.6E-14	6.9E-11	8.2E-09	2.0E-07
18	2.9E-162	1.4E-68	3.8E-38	1.2E-24	1.6E-17	2.5E-13	1.1E-10	6.9E-09
20	3.8E-200	2.0E-84	8.2E-47	4.3E-30	2.9E-21	4.7E-16	9.6E-13	1.6E-10
22	5.1E-242	5.7E-102	2.2E-56	4.1E-36	2.1E-25	4.5E-19	4.9E-15	2.6E-12
24	7.0E-288	3.6E-121	7.0E-67	1.0E-42	6.3E-30	2.3E-22	1.5E-17	2.7E-14
26	0	4.8E-142	2.8E-78	6.9E-50	7.5E-35	5.8E-26	2.8E-20	1.9E-16
28		1.4E-164	1.4E-90	1.2E-57	3.6E-40	7.8E-30	3.1E-23	9.2E-19
30		8.6E-189	8.0E-104	5.9E-66	7.0E-46	5.3E-34	2.1E-26	3.0E-21
32		1.1E-214	5.8E-118	7.6E-75	5.5E-52	1.9E-38	8.8E-30	6.5E-24
34		3.2E-242	5.2E-133	2.6E-84	1.7E-58	3.5E-43	2.2E-33	9.5E-27
36		1.9E-271	5.6E-149	2.4E-94	2.2E-65	3.3E-48	3.3E-37	9.3E-30
38		2.5E-302	7.5E-166	5.7E-105	1.1E-72	1.6E-53	3.0E-41	6.2E-33
40		0	1.2E-183	3.7E-116	2.4E-80	4.1E-59	1.6E-45	2.8E-36
42			2.4E-202	6.4E-128	2.0E-88	5.3E-65	5.4E-50	8.3E-40
44			5.9E-222	2.9E-140	6.8E-97	3.6E-71	1.1E-54	1.7E-43
46			1.8E-242	3.6E-153	9.3E-106	1.2E-77	1.3E-59	2.3E-47
48			6.5E-264	1.2E-166	5.1E-115	2.2E-84	9.8E-65	2.1E-51
50			2.9E-286	1.0E-180	1.2E-124	2.1E-91	4.4E-70	1.3E-55
52			0	2.4E-195	1.0E-134	9.9E-99	1.2E-75	5.4E-60
54				1.5E-210	3.8E-145	2.5E-106	1.9E-81	1.5E-64
56				2.5E-226	5.5E-156	3.1E-114	1.9E-87	2.9E-69
58				0	0	0	0	0

Table 3. Maximum Error Outside of Support for Original Vector

k	1	2	3	4	5	6	7	8
Support *n*×*n*								
0	5.2E-01	7.4E-01	8.8E-01	9.3E-01	9.5E-01	9.6E-01	9.7E-01	9.8E-01
2	5.1E-03	5.4E-02	6.1E-02	4.9E-02	3.8E-02	3.0E-02	2.4E-02	1.9E-02
4	2.4E-05	1.9E-04	6.5E-03	1.3E-02	1.6E-02	1.6E-02	1.5E-02	1.4E-02
6	2.0E-08	2.7E-04	1.2E-03	3.7E-05	2.8E-03	5.2E-03	6.6E-03	7.2E-03
8	1.4E-09	2.3E-05	4.6E-05	5.7E-04	4.8E-04	4.5E-04	1.6E-03	2.7E-03
10	1.2E-11	4.4E-07	2.9E-05	3.3E-05	3.1E-04	4.0E-04	9.6E-05	4.6E-04
12	5.2E-14	9.6E-08	1.1E-06	2.4E-05	2.7E-05	1.9E-04	2.9E-04	2.0E-04
14	1.7E-16	1.2E-08	5.8E-07	2.9E-06	1.9E-05	2.2E-05	1.3E-04	2.1E-04
16	4.2E-17	4.8E-10	6.9E-08	1.0E-06	6.1E-06	1.4E-05	1.8E-05	8.8E-05
18	1.0E-16	3.0E-11	8.7E-09	2.0E-07	2.7E-07	7.5E-06	1.0E-05	1.5E-05
20	8.0E-17	6.2E-12	2.4E-09	4.0E-08	5.6E-07	1.1E-06	7.7E-06	7.1E-06
22	8.2E-17	3.9E-13	3.5E-11	1.3E-08	9.6E-08	5.1E-07	2.3E-06	7.2E-06
24	7.3E-17	4.0E-15	6.1E-11	1.3E-09	2.6E-08	3.1E-07	7.0E-08	3.1E-06
26	7.7E-17	3.2E-15	3.8E-12	7.2E-10	1.4E-08	5.2E-08	3.8E-07	5.2E-07
28	6.4E-18	2.2E-16	1.3E-12	2.4E-11	7.3E-10	1.8E-08	1.8E-07	2.7E-07
30	2.1E-17	2.0E-16	1.9E-13	3.9E-11	1.0E-09	1.3E-08	3.0E-08	2.6E-07
32	4.1E-17	2.5E-17	1.7E-14	1.1E-12	2.8E-10	2.5E-09	1.4E-08	1.1E-07
34	7.9E-17	3.9E-16	6.4E-15	2.0E-12	2.7E-11	6.7E-10	1.2E-08	1.9E-08
36	4.8E-17	1.3E-16	2.9E-16	1.8E-13	3.0E-11	5.6E-10	3.7E-09	1.0E-08
38	1.7E-16	6.4E-17	1.5E-16	9.6E-14	4.3E-12	1.2E-10	1.9E-12	1.0E-08
40	6.8E-17	6.0E-19	5.8E-17	1.5E-14	1.7E-12	2.4E-11	5.9E-10	4.3E-09
42	1.1E-16	6.8E-17	1.4E-16	4.1E-15	7.5E-13	2.4E-11	3.0E-10	6.9E-10
44	4.3E-17	2.7E-17	2.8E-16	9.2E-16	1.2E-14	5.9E-12	5.6E-11	4.1E-10
46	1.0E-16	1.7E-17	9.1E-17	1.0E-16	6.3E-14	7.9E-13	2.0E-11	3.9E-10
48	7.1E-17	1.0E-16	5.7E-17	3.3E-16	1.5E-14	1.0E-12	1.9E-11	1.7E-10
50	6.4E-17	1.1E-16	3.8E-16	6.7E-17	2.3E-15	2.8E-13	6.5E-12	2.7E-11
52	2.2E-16	1.3E-17	3.1E-16	3.3E-17	1.8E-15	2.3E-14	2.3E-13	1.6E-11
54	1.5E-16	1.1E-16	7.0E-17	1.5E-16	4.6E-17	4.2E-14	9.3E-13	1.5E-11

Table 4. Maximum Error Outside of Support for Dual Vector

The values at the wandering lines are summarized below in table 5. The total column is taken from table 2, and represents the total support needed for perfect reconstruction to the accuracy of the variables used, or in other words, the support needed to achieve dimension invariance, or each frame (k value) tested.

	Frame Vector	Dual Vector	Total
k			
1	5	7	12
2	7	17	24
3	9	25	34
4	13	29	42
5	15	37	52
6	17	41	58
7	19	49	68
8	21	57	78

Table 5. $n \times n$ Support Needed for Perfect Reconstruction

CHAPTER 7. Conclusions and Future Study

7.1 Conclusions

Image representation and reconstruction using $2D$ frames of translates are studied in this thesis. An application of the study is in image reconstruction from two or more images of different foci. A major issue of the study is that while the numerical complexity of calculation is almost intractable, the use the dimension invariance of compactly supported frame members makes the calculation possible in such frame systems. The study finds that frame systems with members that approximate compact support are also subject to this treatment.

7.2 Application

Before a practical application of this theory is possible, there are a number of areas that will need further research. Few images consist of objects at exactly two focus distances. This paper can be extended to more than two sets of frame vectors. Even with the use of a smaller dimension space for calculating pseudo-inverses, some of the calculations take an impracticably long time. This problem may be surmounted by pre-computing the duals for fixed σ values and possibly interpolating for intermediate values. There are also engineering and design obstacles to producing multiple images that are properly aligned, and to determining the actual frame vectors.

APPENDIX A, Calculation of the Pseudo-Inverse

Because the frames used in this paper are finite dimensional, the standard dual frame can be found by computing the Moore-Penrose pseudo-inverse. The pseudo-inverse of a matrix A can be calculated using the singular decomposition. Given that $A = U\Sigma V^T$, where U and V are square matrices, and Σ is a diagonal matrix whose diagonal values are the singular values of A, the pseudo-inverse may be calculated as $A^\dagger = V\Sigma^{-1}U^T$.[8]
Once the V^T, Σ, and U are known, it is trivial to find V, Σ^{-1} and U^T, and therefore $A^\dagger$.
A strategy for finding the singular values of A requires computing the positive square roots of the eigen-values of A*A and AA*.[9] Many methods exist for estimating eigen-values exist, see for example Fundamentals of Matrix Computations, 2nd Edition, David Watkins, Wiley Inter-Science, Chapters 5 and 6.

The singular value decomposition calculation was performed using a software package called newmat11 available at http://www.robertnz.net/nm11x.htm. This is a general purpose C++ matrix handling software package. The single value decomposition routine has the following comment with respect to its algorithm, "from Wilkinson and Reinsch: Handbook of Automatic Computation."

APPENDIX B, Software Description

All software used in this project was written by the author, except as noted. This software is available for inspection at ftp://www.schoenbrun.com/pub/msthesis.tar.gz. The software was written using the C and C++ languages and compiled using the 2.95.3 version of the GNU compiler. The operating system used was QNX 6.3.0, which is a Unix like Posix conforming operating system, however no operating system specific features were used in the calculations. The hardware used was a Superserver brand computer model 8042, using 4, 1.5Ghz Intel Pentium IV Xeon processors. None of the software used multiple processors simultaneously. Their availability did help to speed processing during some of the longer tests by allowing multiple simultaneous trials.
For this investigation black and white digital images of dimension 512 x 384 were used. Each pixel is represented by an integer [0..255] where 0 represents pure black, and 255 represents pure white. Color images are typically represented using three RGB (red, green, blue) integers so that the results may be extended by applying the computations to each color channel separately. Each pixel is considered a coordinate in an m x n vector. For computations, the coordinates were represented by an IEEE-754 double precision floating point value. This representation consists of:[10]

Field	Binary Bits	
Sign	1	[0=+, 1=-]
Exponent	11	[-1023,1024]
Mantissa	52	[0,1], approximately 16 decimal digits

The main software modules consist of four C++ classes known respectively as *gaussian*, *image*, *pinverse*, and *sequence*. C++ classes are prototypes of objects that consist of both data and programming code. The data and code are hidden within the object except through its public view. This gives an object the properties of a black box. The user of the object knows only what the object does, but not how it does it.

Gaussian: This class creates a single 2-dimensional prototype vector of specific width and sigma value. The creation code automatically normalizes the data. Sigma values were specified using an index [0..19] which is translated into a real value using the formula .15 + .18 * Index. Internally the data for this vector is stored as a matrix of values, each of type "double". Two objects were created from this class for calculations. The width is always an odd integer so that the center of the function is located at the center of a pixel.

Image: This class was used to hide the details of the .BMP disk image file format. The class was used to read images, provide access to their pixel data, and to save new or modified images. While the image format only supports pixels with integer values from 0 to 255, this class stored and calculated all intermediate data using a floating point type "double". This class also contained the code for creating the foreground and background focused image versions.

Pinverse: This class was used to calculate the pseudo-inverses given two Gaussian objects. The process of calculating the inverse was the most time consuming part of the calculation, at times taking 2 or 3 days. To keep from having to repeat any such calculations, this class automatically stored results in a disk file, and avoided recalculation when a disk version was found. This class also contained the reconstruction code used when the inverse matrix was calculated at full size. The routine that computes single value decomposition for determining the pseudo-inverse is part of the package newmat11 described in Appendix A.

Sequence: Since the frames used in this project consisted of two prototype vectors, it was only necessary to store a pair of dual vectors. This pair was called a *sequence*. This class requires the *pinverse* class for computing these sequences, and then stores them in a disk file. This class also contains the reconstruction code for use when the duals are calculated with a matrix smaller than the original image.

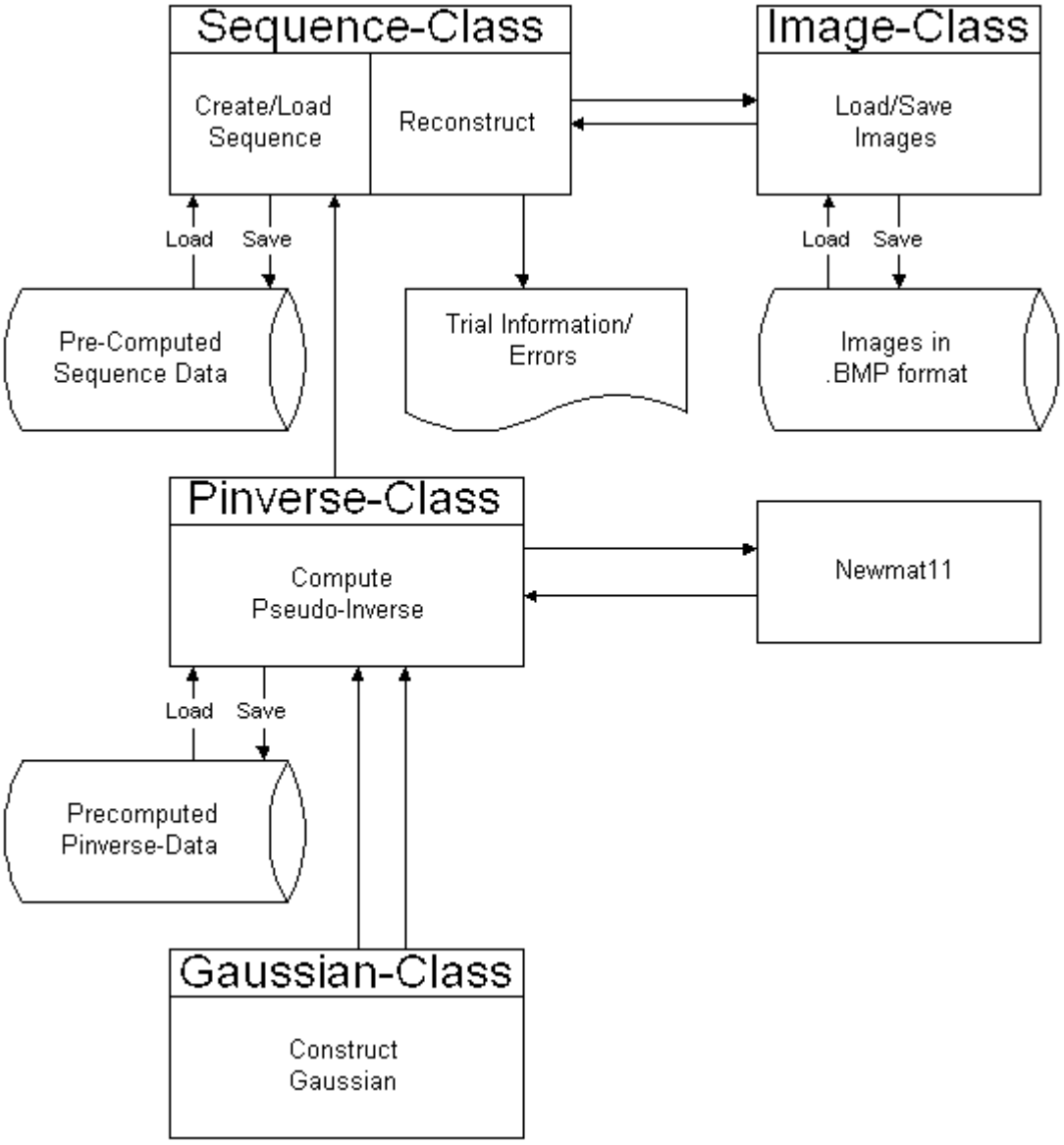

Figure 5. Software Flowchart

These modules were used in a program called reconstruct, which performed the following steps.

1) Create the needed Gaussians.
2) Create the foreground and background images.
3) Create the needed pseudo-inverse
4) Reconstruct the original image
5) Calculate errors between the original and reconstructed images

The average error was calculated as follows

$$ae = \sum_{i=0}^{m} \sum_{j=0}^{n} \left| P_o(i, j) - P_r(i, j) \right| \Big/ mn$$

where P_o is the original image data, and P_r is the reconstructed data.

A GUI utility program called *mapper* was used to manually generate the focus map. This program allowed the user to choose pixels using the original image as a background guide. The GUI used by this software is native to the QNX operating system, and so unfortunately cannot be easily ported to other systems.

APPENDIX C, Optical Description

A photographic camera consists of an optical system, and an image detector, commonly film, or in the case of a digital camera, a charge coupled device (CCD). In an ideal camera, image capture is a linear projection from objects in the three dimensional real world space onto the image detector. In reality, there are a number of distortions that take place. The main distortion is that of focus. A lens works by bending the light path of the original image. This bending is due to the difference of index of refraction of the two media, air and glass or plastic that the light passes through.

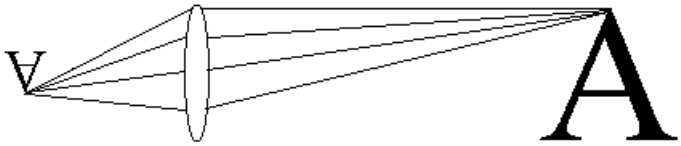

Figure 6. In Focus Image

If different light paths from the same point in the original image do not intersect in the same location on the image detector, so the image will appear out of focus.

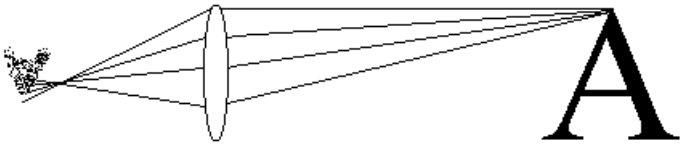

Figure 7. Out of Focus Image

Focus is also affected by the size of the lens opening, referred to as the *f*-stop. A smaller opening or *f*-stop will create an image that is more in focus. In photographic terminology this is referred to as a greater depth of field.

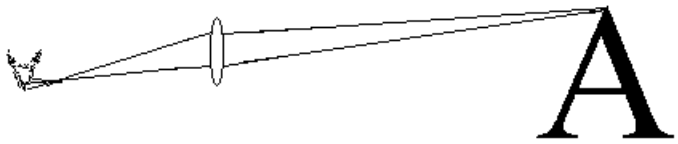

Figure 8. Reduced *f*-stop

It is not possible for all parts of the original image at different distances from the lens to all be in focus, although this can be mitigated by using large *f*-stops.

For this paper the defocusing of an image is simulated by a 2-dimensional Gaussian function. This function has the features of being radially symmetric, and center weighted with an exponential drop off of distance from the center. This of course is not an exact function for lenses in general, but it served as an adequate tool for the investigation. For a practical application, the transformation function of a lens would have to be accurately measured. The techniques for doing this are beyond the scope of this paper. It is also assumed that all the objects in the original image are at exactly two different distances, d_I the in-focus distance, and d_O the out of focus distance. The projection of the original image onto the detector is a function $h:X \rightarrow Y$ where X is the 3 dimensional real world space and Y is the 2 dimensional image detector space. Since all of the objects in the original space are located at only two different z distances from the lens, the function can be rewritten as follows, reducing the problem to one of 2 dimensions:

$$h(x,y,z) = m_I(x,y)h_I(x,y) + m_O(x,y)h_O(x,y)$$

where h_I is an in-focus transformation, h_O is an out-of-focus transformation, and where m_I and m_O are the mapping functions

$$m_I = \begin{cases} 1 \text{ if } z=d_I \\ 0 \text{ if } z=d_O \end{cases} \quad m_O = \begin{cases} 0 \text{ if } z=d_I \\ 1 \text{ if } z=d_O \end{cases}$$

This can further be refined by noting that $m_O(x,y)= 1 - m_I(x,y)$. Also note that h_I should be very close to a simple dilation and a rotation. By using the appropriately transformed coordinates, h_I becomes an approximation of the mathematical identity.

For simplicity an existing digital image is used as a surrogate for the real world space, with an artificially generated mapping function. The transformations with the following convolution

$$R * G(\sigma) = \iint_X R(\tau,\upsilon) G_\sigma(x-\tau, y-\upsilon) d\tau\, d\upsilon = C \iint_X R(\tau,\upsilon) e^{\frac{-((x-\tau)^2+(y-\upsilon)^2)}{\sigma^2}} d\tau\, d\upsilon$$

where R(x,y) is the function associated with the real world image, and C a normalizing constant such that

$$C \iint_X e^{\frac{-(\tau^2+\upsilon^2)}{\sigma^2}} d\tau\, d\upsilon = 1$$

with σ_I and σ_O constants that determine the degree to which a transformation defocuses the original image.

The values of σ are limited to $\sigma_k = .15 + .18k$ where k=[0..19]. This provides a spread of *h* functions from a near perfect focus to significantly out of focus. To model real digital photos, the R function is represented by an m×n matrix,

$[R_{x,y}]$ and h_I and h_O become

$$h_I(x,y) = C_I \sum_{x,y} R_{x,y} e^{\frac{-((m-x)^2+(n-y)^2)}{\sigma_I^2}}, \quad h_O(x,y) = C_O \sum_{x,y} R_{x,y} e^{\frac{-((m-x)^2+(n-y)^2)}{\sigma_O^2}}$$

with the sums over integers [0,…,m-1] and [0,…,n-1]. There is one more important, although somewhat unrealistic simplification in this model.

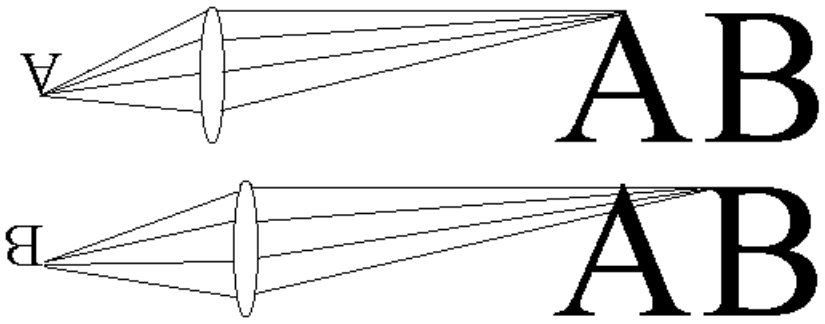

Figure 9. Multiple Foci

In the upper image is an in-focus function h_I mapped to the letter A, and an out of focus function h_O mapped to the letter B. In the lower image is the opposite, an in focus function h_I' mapped to the letter B, and an out of focus function h_O' mapped to A.

It should be clear that $h_I = h_I'$ since they are both close to identities, however in general $h_O \neq h_O'$. If the distance from the letters to the lens is large compared to their relative distances $h_O \approx h_O'$. The simplification $h_O = h_O'$ is used so that the frame will be made of two sets of vectors instead of four. While this is a reasonable simplification for this paper, it may not be valid for practical applications that will need all four functions.

APPENDIX D, A Real World Test

During the course of this project a real world example was tried. A scene with foreground and background objects was set up and two photographs were taken, one focused in the foreground and a second focused in the background. A focus map was calculated using a very simple algorithm. The idea behind this algorithm is that if an out of focus pixel is re-projected using an out of focus frame member, its value should not change, however if it is an in focus pixel it will. The images were reconstructed using the same program as in the thesis tests. The results suffer from a number of real world problems that were not addressed in this paper, however the results are visually interesting.

Image 10. Real world image, background focus

Image 11. Real world image, foreground focus

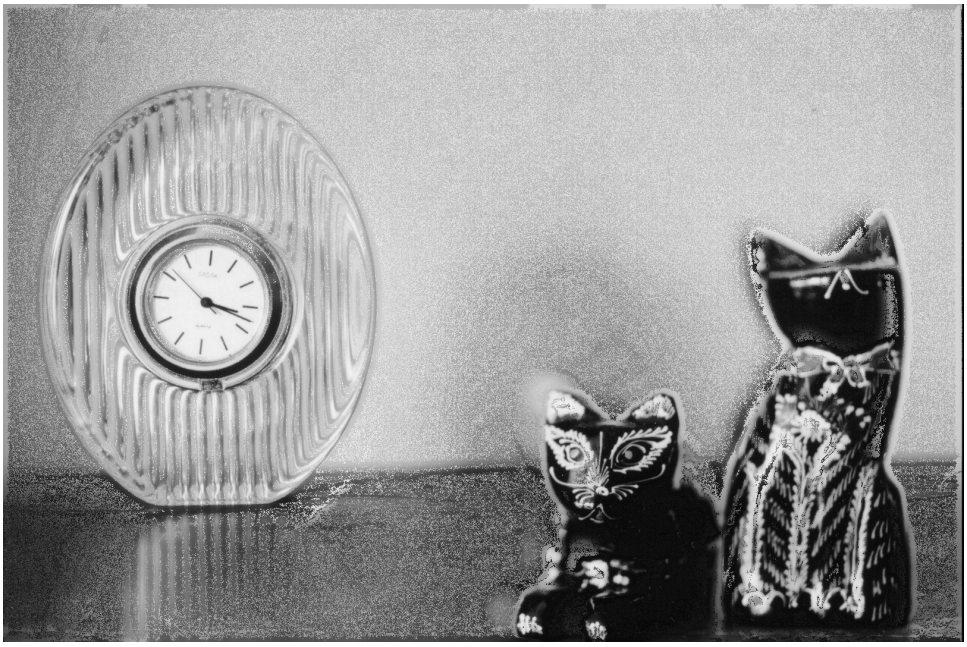

Image 12. Real world image, reconstructed

References

[1] Ole Christensen , An Introduction to Frames and Riesz Bases (Birkhäuser 2003) (Section 1.1 Equation. 1.3, p 3),.
[2] Christensen (Proposition 1.1.2 p 4).
[3] Christensen (Equation 1.6, p 4).
[4] Christensen (Theorem 1.15, Equation 1.10, p 5).
[5] Christensen (Corollary 1.1.6, Equation 1.11, p 7).
[6] Christensen (Section 1.5, Equation 1.21, p 23).
[7] Eugene Charniak, Introduction to Artificial Intelligence (Equation 3.2, p 106), (Addison-Wesley, 1985)
[8] David S. Watkins, Fundamentals of Matrix Computations, 2nd Edition (Wiley Inter-Science, 2002) (The Pseudoinverse, p 277).
[9] Sheldon Axler, Linear Algebra Done Right, 2nd Edition, (Springer, 1997) (Section 7.47, p 157).
[10] http://en.wikipedia.org/wiki/ Double_precision#Double_precision_memory_format