An Empirical Analysis of Super Resolution Techniques

for Image Restoration

by

Jeffrey S. Brown

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 8, 2000

June 2000

Author_____

Department of Electrical Engineering and Computer Science

May 8, 2000

Certified by_____

Prof. Jae S. Lim

Thesis Supervisor

Accepted by_____

Arthur C. Smith

Chairman, Department Committee on Graduate Theses

An Empirical Analysis of Super Resolution Techniques for Image Restoration

by
Jeffrey S. Brown

Submitted to the
Department of Electrical Engineering and Computer Science

May 8, 2000

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Electrical Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

In image restoration, super resolution techniques show promise to create more accurate results than traditional processing methods such as interpolation and inverse filtering. Super resolution techniques use nonlinear information, stochastic information, and information from multiple offset pictures of the same scene to accurately reconstruct images beyond the traditional limits imposed by aliasing and low pass filtering. Unfortunately, there is not a coherent theory of super resolution, nor is there any known comparison of existing super resolution and traditional algorithms. This thesis contains an empirical comparison of three super resolution techniques. These algorithms significantly outperform basic interpolations and inverse filters when multiple offset images of a scene are available. The algorithms' capabilities suggest changes to imaging system design: it is shown that capturing many offset images of a scene can lead to more accurate reconstructions, even if the multiple images come at the expense of lower signal-to-noise ratios.

Thesis Supervisor: Prof. Jae S. Lim
Title: Professor of Electrical Engineering and Computer Science

VI-A Supervisor: Larry Candell
Title: Group Leader, Advanced Space Systems and Concepts

Acknowledgements

To Larry Candell for helping define a topic that was fascinating and personally compelling, and for his subsequent support –

To Prof. Jae Lim for his helpful comments as advisor –

To Edward Wack for providing so much data and furnishing helpful information about algorithms, analysis, and general imaging facts –

To Ed Leonard for his words of wisdom –

To Jennifer Bautista for the support –

To Bill Vanderson for his technical proofreading services –

To my parents for 22 years of encouragement, and for sending me to all the best schools –

Table of Contents

List of Figures

Chapter 1 - Introduction

The pictures created by digital imagers are not perfect representations of the original scenes. Instead, the optics and electronics of the imager distort the true signal, and discrete samples of this distorted signal are produced. The amount of distortion depends on the size of the optics, the size of the photodetecting elements, and the accuracy and tolerances of all components and their support circuitry. If imaging technology were more advanced, the raw sample values might convey sufficiently accurate information about the scene, and no additional processing would be required. With current technology, however, the distortion can be severe and can limit the usefulness of the data.

In modern space-based imagers, two problems are most severe: the low-pass filtering effect of the optics and the aliasing caused by an insufficient spatial sampling rate. The first problem is a consequence of finite aperture size – a finite aperture can only detect a subset of the diffracted light, and this limits the frequencies passed by the optical system. The second problem is due to the spacing of detector pixels on the focal plane. Fabrication technology and physics limit the detector sizes and spacing, which limit the spatial sampling rate. In most cases, current technology does not allow sampling at twice the highest focal plane frequency, so the sample values are corrupted by aliasing.

Since the physical parameters of imaging systems change slowly, scientists and engineers use signal processing techniques to increase imaging accuracy. In the past, linear filters were used, with techniques such as Wiener filtering drawing from the rich history of linear system theory. Unfortunately, traditional linear processing methods are inadequate for processing many modern digital images. While they can mitigate certain transfer function effects and noise, linear methods cannot undo aliasing or cutoff phenomena. Any given frequency may be cutoff by the optics, may be aliased by the discrete sampling (if it is higher than one half of the sampling rate), or may be passed by both systems with some modulation. Even if the frequency is passed with modulation, higher frequencies may interfere via aliasing. In all of these cases, a linear filter cannot reconstruct the original information at the frequency of interest.

Super resolution techniques attempt to restore image information at frequencies beyond the traditional limits imposed by aliasing or low-pass filtering. This goal is impossible given a single set of samples and no *a priori* information. In this case, frequencies that have been eliminated by low pass filtering are irretrievably lost, and frequencies corrupted by aliasing cannot be de-aliased. By incorporating additional information into the restoration, recovering higher frequencies may still be difficult, but is no longer theoretically prohibited. Super resolution algorithms use this knowledge to produce more accurate images.

Super resolution is possible because imaging systems are rich with additional information. For example, images are always of finite size and have nonnegative intensity (due to the physical imaging system). Furthermore, situations exist where multiple images of the same scene are available, with each image offset some non-integer number of pixels. Any such source of additional information restricts the set of candidate solutions, and allows the true image to be estimated more finely and accurately. Chapter 2 discusses specific examples of how this additional information can be used to improve reconstructions.

Many super resolution algorithms have been demonstrated, but there has been no known attempt to compare super resolution algorithms in a general context. Most algorithms have been presented as *ad hoc* solutions to specific imaging situations; rarely have their presentations been accompanied by strict comparisons to other restoration schemes. The most common "proof" of effectiveness is a before-and-after display of a single image. Without any comparison data, it is difficult to find the most effective algorithm for any given imaging situation. There is no principled way to choose among algorithms.

This thesis provides an empirical analysis of super resolution performance. Three common and prominent super resolution approaches have been chosen which are representative of the field: a minimum mean square error (MMSE) method, a maximum a posteriori method (MAP), and a projection onto convex sets method (POCS). These algorithms are described with a common notation and then rigorously tested with a variety of images and parameters (e.g. noise levels, multiple frame offsets). The results are compared both mathematically and subjectively, as described in Chapter 4. This process has created a complete data set that is useful for analyzing imaging systems and choosing algorithms for specific applications.

The remainder of this introduction presents the problem, the motivation, and a more complete description of its solution. Chapter 2 provides a physical and mathematical description of imaging, along with the methods and history of the super resolution field. Chapter 3 gives detailed descriptions of the three major algorithms and their implementations. Chapter 4 covers the evaluation process and lists the test situations and results. Finally, Chapter 5 summarizes the efforts and provides suggestions for future work.

1.1 Problem & Motivation

There are several outstanding problems related to super resolution, but the most compelling is the issue of algorithm comparison. Many algorithms have been demonstrated, but there exists no general consensus regarding which are superior. The lack of an all-encompassing theoretical development makes the algorithms difficult to evaluate and compare to each other. Exacerbating the problem further are the fundamental differences of the methods, and the fact that there is no cohesive framework for describing and analyzing these mostly nonlinear iterative algorithms. The result is that there has been no known attempt to compare super resolution algorithms in any context.

This problem demands attention because more accurate information is valuable both scientifically and commercially. Powerful desktop computers are increasing scientists' processing ability, which similarly increases their demand for more detailed information. Fortunately, the rapid increase in computational power is also promoting the viability of complex restoration algorithms. Many super resolution algorithms employ iterative optimizations which until recently were not computationally feasible. Furthermore, the cost of additional processing is now significantly less than that of building an enlarged aperture imaging system.

A specific problem at MIT Lincoln Laboratory has provided the basis for this inquiry. Lincoln Laboratory supports satellite-based imagers, such as the Geostationary Operation Environmental Satellite (GOES) family of weather imagers. Many satellites are already in orbit, containing fixed optics and electronics that limit their resolution. Discussions with meteorologists have determined that higher resolution images are desirable, so Lincoln would like to use super resolution algorithms to obtain better performance from existing hardware. Which algorithms

should be chosen? Which provide the most accurate reconstruction for a given situation? How should imaging system parameters be set to take advantage of the algorithms? These questions must be answered before super resolution techniques can be applied appropriately.

If super resolution algorithms prove effective, a set of additional questions becomes relevant. These questions deal with the design of imaging systems to take advantage of the newfound processing ability: Is it better to design an imaging system to take one high signal-to-noise ratio (SNR) image, or several offset images each with a lower SNR? With super resolution techniques, the latter may provide a more accurate final reconstruction. How helpful is it for the frames to be uniformly spread? The same number of frames at random offsets may lead to similar accuracy. Is there an optimal ratio of aperture size to pixel size? The use of a smaller and cheaper objective lens may not harm performance.

To answer these questions, super resolution algorithms must be thoroughly analyzed. To do so requires overcoming two major comparison issues: the choice of evaluation criteria, and the choice of algorithms. The choice of evaluation criteria and metrics is not obvious; there is no consistent measurement used in super resolution literature. In fact, the most common "proof" of an algorithm's effectiveness is a before-and-after display of a few sample images [Alam 1997, Gillette 1995, Irani 1990, Cohen 1998, Cheeseman 1994]. Unfortunately, this method does not facilitate comparison, nor is it necessarily an indication of performance. A reasonable comparison must encompass explicit mathematical comparisons along with the subjective image viewing.

The second interesting issue is the application of the metric. Comparing every super resolution technique ever proposed would be an enormous task. The fact that restoration is an ill-posed problem means that there is no single right answer, and there are limitless reasonable methods [Andrews 1977]. We would like an indication of effectiveness that does not involve coding every algorithm. It would be useful to apply the metrics to a few general algorithms that are representative of common approaches. The performance of these general methods will establish benchmarks and provide a common base for more specific future inquiries. For this approach, the difficulty is in choosing the methods that would be most useful to compare.

1.2 Solution & Thesis Definition

This thesis contains an empirical analysis of three major super resolution techniques. The goal was to specify a fair set of evaluation metrics and use them to compare prominent super resolution approaches (MMSE, MAP, and POCS) in a variety of contexts. This work provided information that is generally applicable to the field, as well as specifically interesting to Lincoln Laboratory. The process also produced a data set that answers the questions posed above.

The comparison was empirical for practical reasons. Most super resolution techniques consist of iterative optimizations, which often include peculiar constraints or non-linear functions. These procedures are difficult to analyze theoretically since they cannot be easily viewed within existing frameworks; they cannot be analyzed as LTI filters, or any similarly established and well-understood method. The algorithms are described mathematically in the thesis, and their basic operation is explored; however, the empirical data is emphasized, since a complete theoretical analysis is beyond the scope of a Master's Thesis.

With these facts in mind, we can discuss the solution to the two problems posed above: the choice of comparison criteria and the choice of algorithms. The choice of comparison criteria comprises both mathematical and subjective portions. Mathematically, no single metric is a universal solution, so we have selected several mathematical quantities to compare. The metrics, which are described in Chapter 4, compare mean squared error and Laplacian mean squared error. The subjective criteria are used as a confirmation and "sanity check" for the mathematical data. They include the viewing of the actual images and the visual identification of small features.

The choice of algorithms concentrated on general-purpose methods. There exists a substantial body of super resolution literature treating a variety of specific imaging situations: stereoscopic perspective changes, multiple independent motions between frames (often video sequences), or searching for a particular target. For simplicity, we chose methods that treat only global translation between frames and do not require other constraints on the original image. The simple case of translational motion is sufficient to display the characteristics and potential of the algorithms. Also, global translation is a good approximation for most satellite-based imagers.

With the chosen algorithms and metrics, an empirical comparison provides the hard data. The test setup is designed to produce several types of degraded data: many simulated images, various

noise levels, and various multiple frame parameters (differing number of frames and offsets). The test includes several "control" restorations, including interpolation and inverse filtering. All results are compared both subjectively and according to the evaluation metrics, with the latter being emphasized.

The desired outcome was a robust set of data concerning super resolution algorithms. This work was designed to provide explicit mathematical results to the question of algorithm comparison; this data is used to answer the questions posed earlier regarding algorithm comparison and imaging system design. The data is useful as a theoretical and practical guide to super resolution techniques.

Chapter 2 - Background

This chapter describes the imaging process, discusses the basis of super resolution, and presents a compact history of the field. The Imaging Models section describes imaging both physically and mathematically. The Principles of Super Resolution section defines super resolution and examines the basis and the soundness of its methods, including a discussion of why super resolution is possible. Finally, the History section recounts a brief history of the field.

Keep in mind throughout that this thesis is pursuing methods of image *restoration*, as opposed to image *enhancement*. Restoration is concerned with mimicing the original scene as accurately as possible, which is often the goal for scientific applications. Enhancement, on the other hand, is willing to sacrifice accuracy in order to enhance certain qualities of the image. The result of enhancements (such as high-pass filtering or histogram equalization) may look "sharper" or "brighter" than the original, but will often be a *less* accurate depiction of the original scene. For a discussion of enhancement verses restoration, see [Lim 1990] or a similar image processing text.

2.1 Imaging Models

The majority of modern digital imaging systems use solid state detectors to convert light intensity into an electrical quantity. A simple diagram in Figure 1 shows the basic components of an imaging system: the true image, optics, photodetectors, and readout electronics. The photodetector may be a two-dimensional array or a linear array. The former case, known as a staring array, can create two-dimensional images without any additional components. The latter case, the linear array, must employ a scanning mirror or similar device to provide the second dimension as the mirror scans across the image plane through time. Both types of arrays can be found in imaging applications; fortunately, both models can be treated similarly as they suffer from the same degradations.

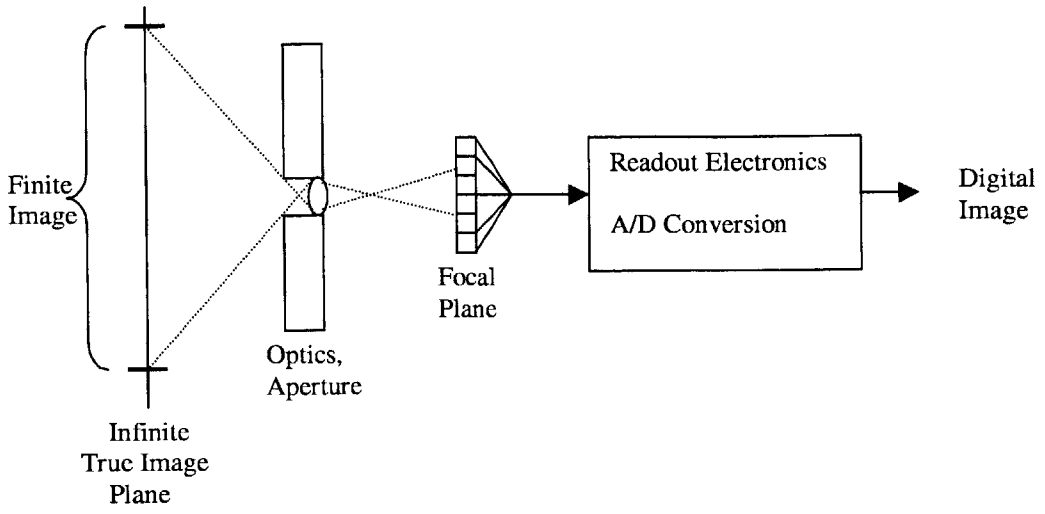


Figure 1 – General Solid State Imaging Model

2.1.1 Degradations

The digital image in Figure 1 provides data about the true image plane, but this data has been altered by several phenomena. Initially, note that only a finite portion of the image plane is within view of the optics, due to the mechanical housing limiting the field of view. The optics then impose a transfer function of low-pass character [Goodman 1968]. The result is an image on the focal plane. The photodetectors cannot perform ideal (delta-function) sampling, but must integrate photons over a finite area; this integration corresponds to another low-pass transfer function. The focal plane is of finite size, which contributes a windowing function. Finally, the charge values may undergo electronic filtering before being converted to digital samples. This entire system is depicted in Figure 2, which is an adaptation of a model from [Alam 1997].

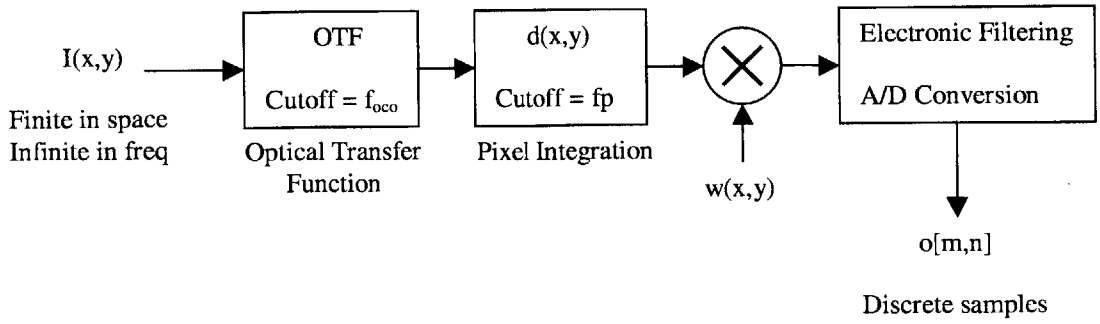


Figure 2 – Transfer functions of a general imaging system

Some definitions and clarifications are helpful before continuing with the degradation discussion. The optical transfer function (OTF) and its inverse Fourier Transform, the point spread function (PSF), are assigned liberal definitions throughout imaging literature; sometimes they refer to the transfer function of the optics only, while sometimes they encompass the effects of the entire imaging system. We will use the strict definition concerning optics, for which diffraction effects cause a hard cutoff frequency, f_{oco}, ($|OTF| = 0$ for $f > f_{oco}$) for a perfect lens [Goodman 1968]. In contrast, we will use the term Spatial Response Function (SRF) to represent the cumulative effect of the PSF, pixel-integration, and any electronic filtering.

The degradations above present a formidable challenge to overcome. Frequencies above f_{oco} are completely eliminated by the OTF. In many cases, frequencies below f_{oco} are corrupted by undersampling. Undersampling is common in space-based imagers and occurs when the detector spacing is greater than $1/2f_{max}$, where f_{max} is the highest spatial frequency on the focal plane. These aliasing and low-pass filtering operations are impossible to reverse without incorporating additional information into the solution and utilizing nonlinear restoration techniques. The super resolution field has developed to solve these difficult problems and to reconstruct information beyond the traditional limits of linear techniques.

2.2 Principles of Super Resolution

It is now possible to discuss the principles and theories of super resolution using the systems and terminology developed above. Super resolution (SR) typically refers to algorithms that increase the sampling rate of an image while also introducing additional frequency content. This distinguishes SR algorithms from most interpolation methods, which sample the existing information more finely, but do not introduce additional information, or do not do so in a principled and justified manner. In short, super resolution results are more accurate than interpolations or linear filters.

Super resolution is difficult to achieve because of the severity of imaging system degradations. The sampling process necessarily requires either low pass filtering or aliasing, and often includes elements of both. These factors create an ill-posed inverse problem: given a single image and no additional information, there are an infinite number of true distributions that could have produced

the observation. This difficulty is most apparent in a frequency domain depiction, as shown in Figure 3. Complexity arises from the relationship of f_{oco}, f_p, and the sampling rate.

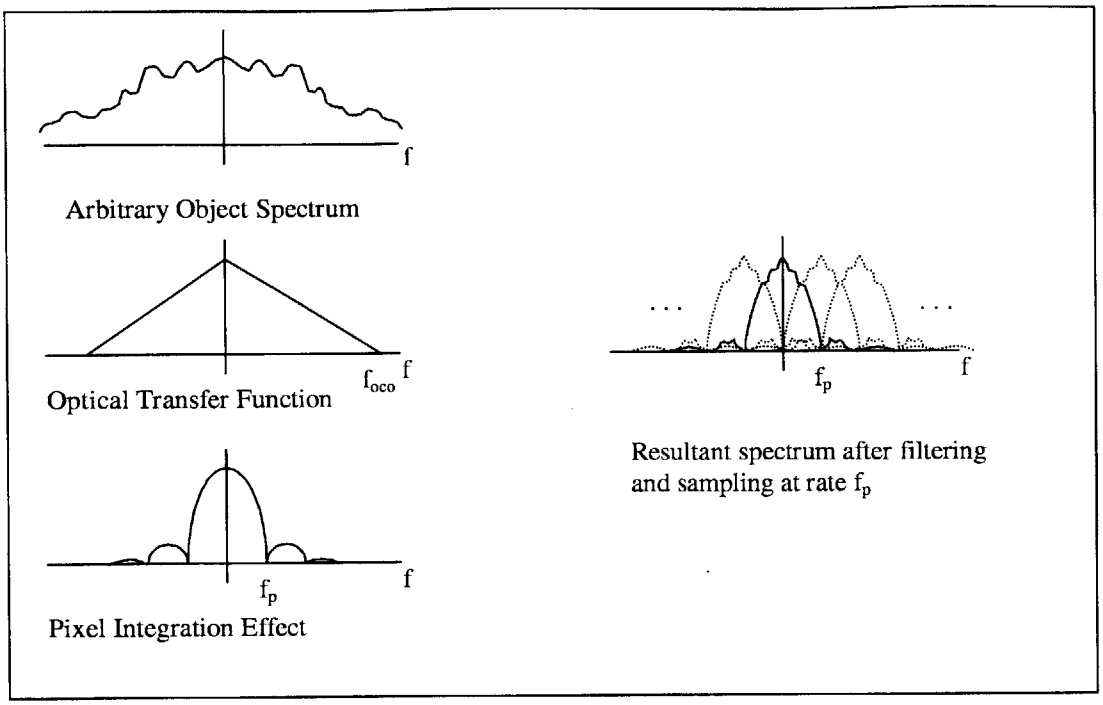
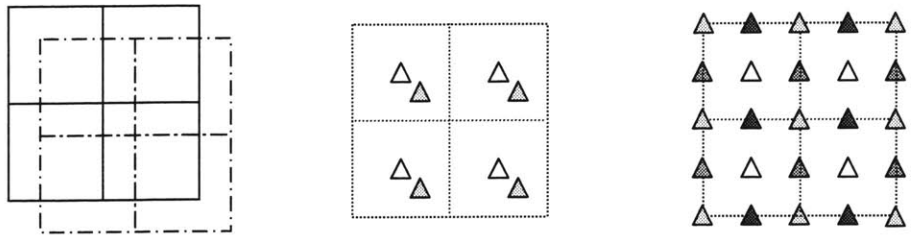


Figure 3 – A frequency domain depiction of imaging. For most modern space-based imagers $f_p < f_{oco}$, as pictured. f_p is also the highest possible spatial sampling rate for many imagers (see Appendix A for a derivation and explanation). In this case, the post-sampling spectrum can be seen to have substantial aliasing.

Despite the difficulties shown in Figure 3, super resolution can be achieved if sufficient additional information is incorporated into the restoration. Fortunately, imaging (particularly space-based imaging) has many deterministic and stochastic information sources. Traditional techniques, such as interpolations or inverse filters, cannot use this information because it is often nonlinear or stochastic. Super resolution techniques excel because they are formulated explicitly to take advantage of this information.

The main source of additional information used by super resolution techniques is data from multiple offset images of the same scene. If these multiple images, or frames, are offset by sub-pixel amounts, the result is a grid, possibly irregular, of more densely spaced samples. A technique called microscanning has been proposed, which creates a uniform grid of samples by purposely controlling these offsets [Watson 1992]. Most super resolution techniques do NOT use

controlled microscanning, however, but rather rely on random offsets introduced by line-of-sight jitter, multiple orbital passes, or other factors. Figure 4 depicts different multi-frame situations. Regardless of the offsets (controlled microscanning or uncontrolled and random), a substantial amount of new data is added by each additional image, and this multi-frame data is utilized by super resolution techniques.



(a) (b) (c)

Figure 4 – Often multiple frames are available at sub-pixel offsets from each other. A two-frame example is shown in (a). The graphic in (b) represents the data from (a) in another format that is more amenable to picturing many frames. Each triangle represents the center of a pixel, and the pixel size is shown as a dotted line for perspective. In (c), we see an example of microscanning, where the array was shifted in ½ pixel increments in each direction, creating an effective sampling grid at twice the original resolution in each direction.

While multi-frame data is important, other information is also used by super resolution techniques to improve restoration accuracy. Two attributes commonly exploited are nonnegativity and finite support. By physical definition, the intensity of any point cannot be negative. Thus any reconstructions with negative values can be modified or eliminated. Likewise, Figure 1 shows that a mechanical housing limits the field of view of the optics, so the image being reconstructed is always of finite size.

In addtion, super resolution techniques use stochastic criteria. Stochastic knowledge can lead to impressive results if the knowledge is specific enough, as demonstrated in a one-dimensional experiment in [Ruderman 1992]. Very specific knowledge is usually unknown, however, and algorithms commonly use just a smoothness criterion that gives smoother images more precedence as solutions. Smoothness constraints are dependent upon the notion that natural scenery is smooth more often than highly oscillatory. This notion is not infallible, and cases exist where the smoothness constraint adversely affects the reconstruction. Nevertheless, smoothness constraints are common and are additionally helpful in suppressing ringing artifacts.

2.2.1 Mathematical Description

The descriptions above are qualitative, but the super resolution problem and solution can also be exhibited mathematically. The imaging problem is posed as

$$\boldsymbol{y} = \mathbf{W}z + \boldsymbol{n} \quad , \tag{1}$$

where $\boldsymbol{y}$ is a column vector of observed pixel values, z is a column vector of true pixel values, $\boldsymbol{W}$ is a projection operator that is directly related to the SRF, and $\boldsymbol{n}$ is an additive noise vector. Suppose $\boldsymbol{y}$ contains one frame's worth of data, and we wish to reconstruct $\mathbf{z}$ at a resolution twice the resolution of $\boldsymbol{y}$ in each direction (four times the resolution area-wise). If N is the number of elements of $\boldsymbol{y}$ (the number of data pixels available), then z has $4N$ elements. The problem is ill-posed, with N equations and $4N$ unknowns. Stated differently, $\boldsymbol{W}$ is a wide matrix with N rows and $4N$ columns. The columns of $\boldsymbol{W}$ cannot be linearly independent, so $\boldsymbol{W}^T\boldsymbol{W}$ is singular and there is no exact solution, even with $\boldsymbol{n} = \boldsymbol{0}$.

Mathematically, the goal of a super resolution algorithm is to regularize this ill-posedness and generate a solution z that is most probable. Additional frames of data help substantially, since each new data point adds a row to $\boldsymbol{W}$. In the above case, four frames of data is technically sufficient to solve the problem exactly, if there were no noise. Other constraints such as nonnegativity also help regularize the problem, but in a less obvious way. They allow certain z's to be eliminated as candidate solutions.

A final note concerns the availability of the data discussed above. It was stated that super resolution algorithms use the imaging model ($\boldsymbol{W}$) and multiple frames of data to produce their super resolved outputs. Unfortunately, an imaging model and the offsets of the multiple frames may not be known exactly *a priori*. For completeness, the calculation, estimation, and use of this information is discussed in this thesis. It can be found in Chapter 4, and then again in the Future Work section of Chapter 5.

2.2.2 Variety of Techniques

With knowledge of the existence and theory of super resolution, there are many different ways to pose and solve the restoration problem. Some algorithms compose a cost function and use iterative optimizations to find minima; some pose the problem stochastically and attempt to maximize a probability distribution; still others use concepts like the theory of projection onto convex sets; and many additional techniques exist, including hybrids of several concepts. No approach is universally superior, and the methods are difficult to compare because they often incorporate different information in unique ways. Also, the results produced in super resolution literature are usually subjective and not suitable for algorithm-to-algorithm comparison.

To demonstrate the breadth of super resolution techniques, a classification of some super resolution algorithms is shown in Figure 5. The components of Figure 5 will be discussed in Chapter 3, but they are shown here to illustrate how numerous and varied the techniques are. Because the differences between algorithms are so large, an empirical comparison is more tractable than a theoretical one. Many of the algorithms from Figure 5 claim superior performance; encouragingly, this thesis confirms those results with mathematical measurements, helping prove that super resolution is a field with impressive potential.

The background and theory of super resolution techniques has now been presented. It is clear that although the image restoration problem is difficult, super resolution algorithms are beginning to use information that has previously not been used in reconstructions. This includes data from multiple frames that are randomly offset from each other as well as the concepts of nonnegativity and finite support. The exact manner of incorporating the information varies widely, and this can be seen in the variety of techniques that have arisen. The techniques are treated rigorously in Chapter 3, which follows the summary of super resolution history below.

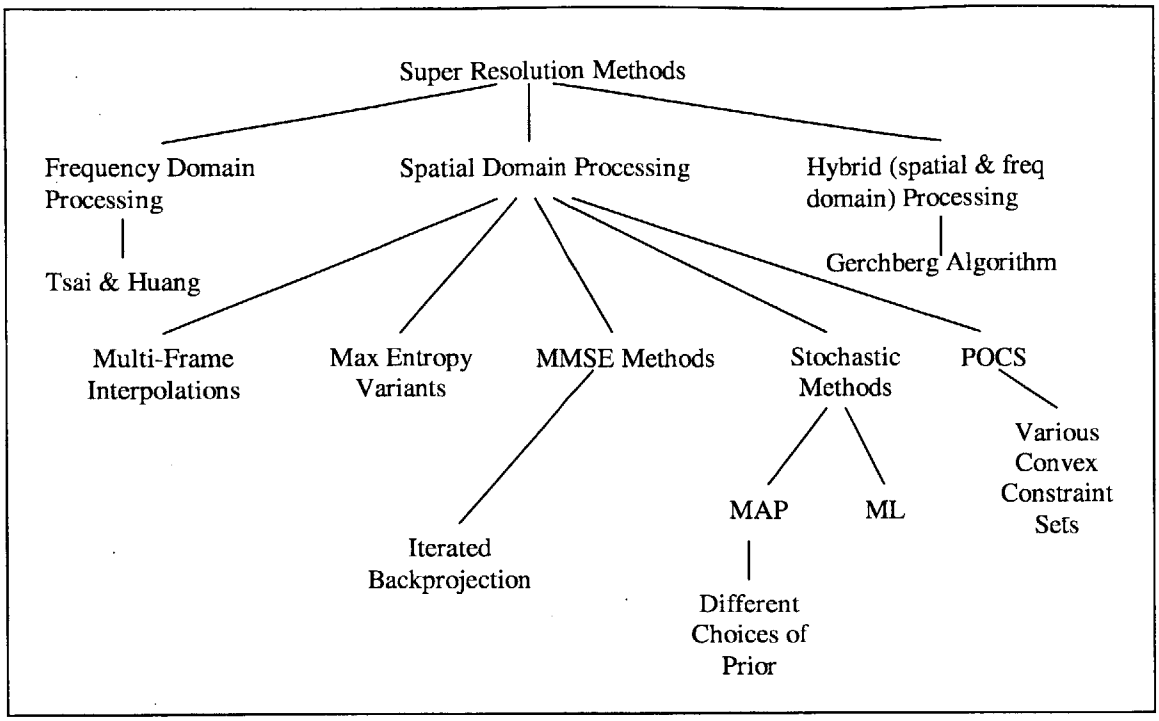


Figure 5 – Classification of some super resolution techniques. Note the wide variety of methods. These methods will be discussed more closely in Chapter 3.

2.3 History of Super Resolution

Super resolution of imagery is a nascent field, with few results before the 1980's. This is not surprising, however, due to the computational power required by most algorithms. Almost all super resolution algorithms use either nonlinear information, non-uniformly spaced samples, or stochastic information. The data can rarely be processed in a straightforward manner, and often requires nonlinear iterative processing. This processing is computationally intensive and was infeasible until recently. After a computational threshold was reached, super resolution algorithms began to appear, and they have proliferated greatly as computational ability has expanded.

The first notions of super resolution of imagery appeared in the 1960s (to the author's best knowledge), and were based not upon multiple frames of data, but upon finite support. These first results assumed oversampled imagers, in which the cutoff frequency of the optics (f_{oco}) was the limiting factor and there was no aliasing. If the truth image has finite support, Harris showed

that without noise, the truth spectrum can be recovered exactly for all frequencies [Harris 1964]. Even with noise, spectrum "extrapolation" can occur beyond f_{oco}, which was shown by algorithms such as Frieden's Maximum Likelihood and Maximum Entropy [Frieden 1972] and Gerchberg's Error Energy Reduction [Gerchberg 1974]. This line of restoration has continued in the 1990s with work by several authors [Hunt 1995, Sementilli 1994], most notably Hunt, who has written an overview of principles and performance in [Hunt 1995].

The techniques above can achieve super resolution, but their usefulness is limited because they require oversampled imagers. Most modern imagers are undersampled, and the aliasing introduced by the undersampling invalidates the principles of those algorithms. For this undersampled case, there were few image processing solutions until 1984, when Tsai and Huang published their seminal multi-frame work [Tsai 1984]. Tsai and Huang used multiple Landsat images from different orbital passes to produce a single image at a higher sampling rate. This is the first known use of multiple offset images to solve the image restoration problem.

Tsai and Huang's work initiated an enormous surge in super resolution development, and by the early 1990s, many new multi-frame algorithms had been proposed. The use of a multiple-frame projection onto convex sets (POCS) algorithm was proposed in 1989 by Stark and Oskoui [Stark 1989]. A technique called iterated backprojection was proposed by Irani and Peleg in 1990, which minimized a minimum mean squared error criteria with an approach similar to tomography [Irani 1990]. In 1994, Cheeseman *et al* posed the problem stochastically and produced a maximum *a posteriori* solution [Cheeseman 1994]. Subsequently many variants arose from each of these techniques, producing the variety seen today in the field.

The current state of super resolution includes methods for most conceivable imaging situations. This thesis is concerned with uniform translational motion between frames (for reasons already discussed). Nevertheless, algorithms have been developed to use data from multiple spectral bands, stereoscopic perspective changes, multiple independent motions within frames, and other sources. For the case of uniform translational motion, Figure 5 provides an outline of some current techniques, which will be discussed in Chapter 3. The future of super resolution and all its variants is discussed in the Future Work section of Chapter 5.

Chapter 3 - Method Descriptions

Many super resolution algorithms exist, and the goal of this thesis is to analyze several of the most common. In the previous chapter, Figure 5 classified the algorithms of interest: those treating uniform translational motion between frames. Three algorithms (a Minimum Mean Squared Error (MMSE) method, a Maximum a posteriori (MAP) method, and a Projection onto Convex Sets (POCS) method) have been chosen from this list for an empirical comparison and are circled in Figure 6. Each algorithm is described thoroughly in this chapter, and the benchmark methods to which they are compared are given a cursory review.

As discussed in the introduction, we are pursuing prominent super resolution algorithms that represent a wide sampling of the total super resolution field. The particular methods used in this thesis were chosen according to a variety of criteria, including:

- ubiquity of the method in the literature,
- general applicability,
- reasonable implementation,
- lack of explicit training periods or numerous operator-set parameters,
- potential for superior performance.

The ubiquity criterion guarantees relevant results by directing the search away from obscure methods. The general applicability criterion is important because some algorithms have been derived for very specific imagers, SNRs, etc., which we wish to avoid. A reasonable implementation is important for the eventual real-world application of the techniques. The lack of training periods or human-set parameters is a practical matter, so that the algorithms can be operated by users without expert knowledge of the algorithm parameters or training. Finally, the potential for superior performance points us towards algorithms that use substantial *a priori* or stochastic knowledge.

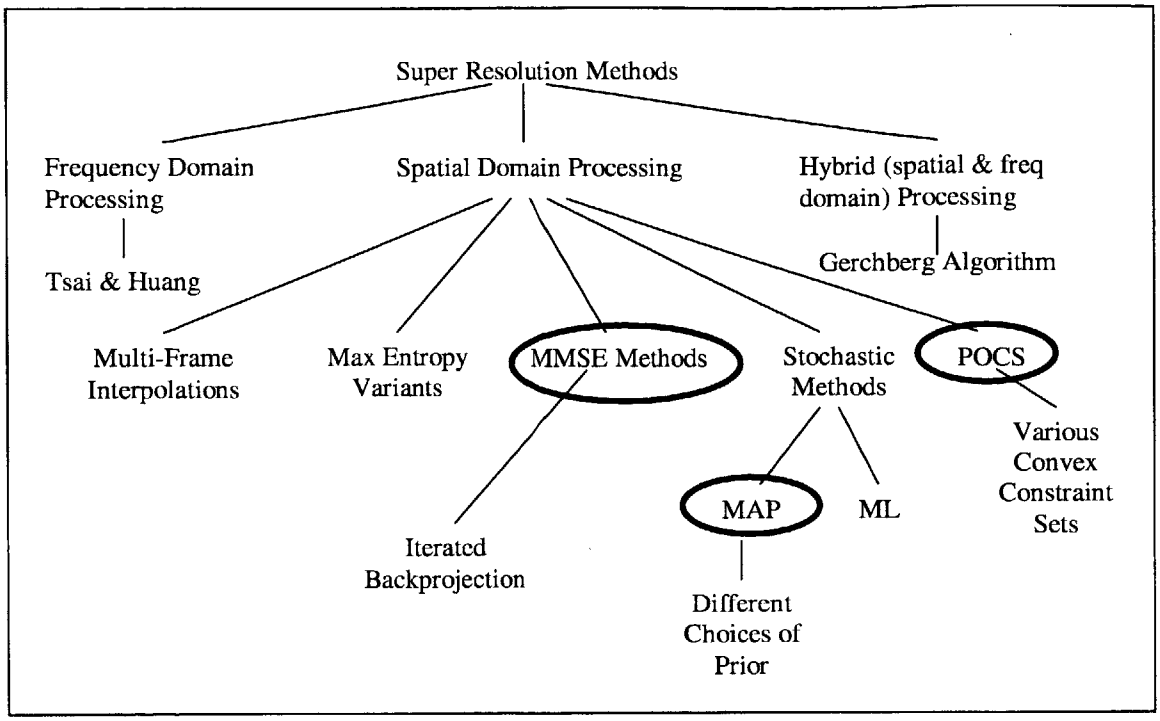


Figure 6 – The super resolution algorithms chosen for comparison are circled.

Unfortunately, even these criteria do not guarantee a *completely* objective choice of algorithms. Some algorithms were surely overlooked during literature searches. The author's biases may also be reflected in the choices. While we strove for objectivity, it's important to remember that these methods represent one author's analysis of the above factors.

After analyzing the above factors, three super resolution methods were chosen for comparison: a Minimum Mean Squared Error (MMSE) method, a maximum a posteriori (MAP) method, and a projection onto convex sets (POCS) method. It's reasonable to argue that the chosen methods constitute a fair sampling of the super resolution field. The three methods are cross referenced copiously in super resolution literature. Most of the methods can be found both in restoration texts (such as Andrews' and Hunt's *Digital Image Restoration* [Andrews 1977]) and in summary articles (such as "Super-Resolution from Image Sequences – A Review" [Borman 1999] or "Survey of recent developments in digital image restoration" [Sezan May1990]). These algorithms are popular, widespread, and important to the super resolution field.

Maximum Entropy (ME) methods can also be found in the references above, yet they have not been included in this comparison. Of the many ME algorithms in existence, only one could be

found that was derived for an ill-posed reconstruction [Lyon 1997]. Conceptually, this algorithm's operation was similar to the chosen MMSE approach. Experimentally, this was confirmed by results that were also very similar to the MMSE results. Since this particular ME formulation took substantially longer than the MMSE algorithm to produce similar results, it was omitted from the tests.

Two final points are relevant to the descriptions below. The first is that for each general approach, one *particular* implementation was chosen for the description and testing. These implementations were chosen for their clear presentation or ease of realization, and not for speed or efficiency. The goal of this thesis is to identify the methods that create the best reconstructions; once identified, any specific approach can be carefully studied and optimized, employing the references and notes from the Future Work section. The second point is that the descriptions cover only the reconstruction portion of the algorithms. Some literature includes methods for parameter estimation (SRF, SNR, etc.) or multiple-frame offset estimation; since these tasks are peripheral to the real reconstruction work, these portions have been omitted from the descriptions.

3.1 Minimum Mean Squared Error (MMSE) Methods

Given a model of the imaging process, any high resolution image can be passed through the model to produce a simulated low resolution output. In matrix notation, $\boldsymbol{W}$ (the imaging model) is known, so an estimate $\boldsymbol{y_e}$ can be produced from the high resolution image z (a column vector of data points) via:

$$y_e = \boldsymbol{W}z \quad . \tag{2}$$

MMSE methods return the high resolution image for which the simulated output is closest to the actual observed data (in the MMSE sense). Mathematically, the output $\hat{z}$ is given by

$$\hat{z} = \arg\min_{z}(y - y_e)^T (y - y_e) = \arg\min_{z}(y - Wz)^T (y - Wz) \quad , \tag{3}$$

where $\boldsymbol{y}$ is a column vector of the observed pixel values.

This criterion is favorable, but it is also incomplete: the inverse imaging problem is ill-posed, so many high resolution images exist which minimize the error. An additional choice must be made among the qualified candidates.

The first MMSE method to be presented was an iterative backprojection algorithm proposed by Irani and Peleg in 1990 [Irani 1990]. The term "iterative backprojection" refers to the algorithm's operation: a high resolution estimate is composed, the imaging process is simulated, then the differences between the simulated images and the actual images are backprojected to modify the high resolution estimate. This process is iterated, with each successive iteration producing a more accurate high resolution image. The ill-posedness problem is resolved by the choice of weights in the backprojection. By choosing certain weighting structures, certain image qualities (like smoothness) can be given precedence. Unfortunately, this control structure is implicit in the algorithm, and is difficult to understand and utilize effectively.

Many extensions to Irani and Peleg's basic approach have been presented. A simple yet powerful example is a solution by Hardie *et al* that introduces an explicit regularization term [Hardie 1998]. A MMSE term, along with a regularization term, constitute a composite cost function that is minimized by an iterative optimization. Since the regularization term is an explicit cost term, it can easily be understood and modified. The regularization term used below is a linear smoothness constraint. The composite cost function is thus linear, but is still minimized iteratively because of the huge number of variables. This Regularized MMSE (RMMSE) method was coded for the algorithm comparison, and a complete description is given below.

3.1.1 Technical Description

Hardie *et al*'s RMMSE algorithm seeks the high resolution image **z** that minimizes a cost function, C(**z**):

$$\hat{z} = \arg\min_{z} C(z) \quad . \tag{5}$$

The cost function is defined as:

$$C(z) = \frac{1}{2}(y - Wz)^T (y - Wz) + \frac{\lambda}{2}(Az)^T (Az) \quad , \tag{4}$$

where:

- y is a column vector containing the pixel values of all the low resolution data.
- z is a column vector containing the pixel values of the high resolution estimate. The size of z is determined by the resolution increase.
- W is a matrix determined by the system transfer function.
- λ is a scalar parameter that determines the relative weight of the regularization term and typically has a value around 0.05-0.1.
- A is a matrix of weights that produce a regularization term. For instance, if z_a is a given pixel in the high resolution image, and z_b, z_c, z_d, and z_c are the 4 cardinal neighbors of z_a, then *row a* of A would read:

column: b c a d e

[0 ... 0 –¼0 ... 0 –¼ 0 ... 0 1 0 ... 0 –¼ 0 ... 0 –¼ 0 ... 0]

This weighting is used by Hardie *et al* and the regularization term has lower cost for a smooth image than a rapidly varying image.

With the cost function thus defined, it is minimized with a standard iterative optimization (see [Bertsekas 1995] or a similar optimization text for review). This particular gradient descent method operates in the following manner:

- the gradient of the cost function is calculated
- the line extending in the negative gradient direction is searched for lowest cost (line minimization method)
- z is updated to this point of lowest cost
- the next iteration begins

The initial value of z used to start the iteration was chosen to be the composite created by averaging all low resolution frames. The calculations are straightforward but algebraically

intensive. The results, as derived by Hardie *et al*, are given below [Hardie 1998]. The iterative update procedure is tersely written as

$$\hat{z}^{n+1} = \hat{z}^n - \varepsilon^n g^n \quad , \tag{5}$$

where the gradient, $\boldsymbol{g}^n$, is given by

$$g^n = W^T(Wz - y) + \lambda A^T A z \ . \tag{6}$$

We calculate ε by minimizing $C(z^n - \varepsilon^n g^n)$ with respect to ε, yielding:

$$\varepsilon^n = \frac{(Wg^n)^T(Wz - y) + \lambda(Ag^n)^T Az}{(Wg^n)^T(Wg^n) + \lambda(Ag^n)^T(Ag^n)} \ . \tag{7}$$

The above derivation is fairly general: the PSF need not be space-invariant, nor must it remain constant from frame to frame. For this study, we're assuming a space-invariant PSF, which allows some significant simplifications and computational savings. These computational efficiencies can be found in the Matlab code.

This RMMSE algorithm was implemented directly from the above equations. The Matlab function that implements the algorithm is located in Appendix B, which contains the code for all the algorithms in this chapter.

3.2 Maximum *a posteriori* methods

The maximum *a posteriori* (MAP) approach produces the most probable high resolution reconstruction, given the sample values. This is accomplished by creating a stochastic model of the entire imaging system, which includes stochastic characterizations of all noise sources as well as the truth image. Once the model is created, the algorithm produces the high resolution (HR)

restoration that is most probable. Under some conditions (e.g. Gaussian noise), the MAP approach is similar to the MMSE approach.

A MAP algorithm chooses the high resolution image $\hat{z}$ that is most probable given the data points, the imaging model, and the various probability distribution functions (PDFs):

$$\hat{z} = \arg\max_{z} \Pr\{z \mid y\} \ , \tag{8}$$

where y is a column vector of low resolution data points. Using Bayes's theorem, this can be rewritten as

$$\hat{z} = \arg\max_{z} \Pr\{y \mid z\} \Pr\{z\} \ . \tag{9}$$

Since the logarithm is an order preserving function, the maximum can also be found via

$$\hat{z} = \arg\max_{z} \ln\left(\Pr\{y \mid z\} \Pr\{z\}\right) = \arg\max_{z} \left(\ln \Pr\{y \mid z\} + \ln \Pr\{z\}\right) \ . \tag{10}$$

The first term of the sum, $\ln \Pr\{y \mid z\}$, is the likelihood term, which is maximized by Maximum Likelihood (ML) algorithms. This term represents how well the data correspond to the HR estimate. If the restoration is ill-posed (i.e. more unknowns than data points), there may be many $\hat{z}$'s that are all maxima. If a ML approach produces a single solution, that solution is suspect: the algorithm is making a choice among equally qualified candidates, but the mechanisms of the choice are not explicit – they are hidden within the algorithm or the initial conditions. In this ill-posed case, the prior term, $\ln \Pr\{z\}$, makes MAP a better choice. The prior term explicitly represents *a priori* notions of the data to be observed.

The choice of prior term affects the final output of the algorithm. Many priors use notions of smoothness, since many objects, both natural and man-made, are more likely to be smooth than highly oscillatory. While they may fail, especially under contrived special cases, they are often beneficial and certainly the most ubiquitous. One of the first MAP multi-frame algorithms, by Cheeseman *et al* [Cheeseman 1994], uses a smoothness prior in which each HR pixel's value is

modeled as a normal distribution. Hardie *et al* also propose a solution using a Gaussian prior [Hardie 1997].

Although the Gaussian prior is popular, it was not used for this algorithm comparison. It is conceptually similar to the smoothness constraint of the RMMSE algorithm, and it doesn't demonstrate the flexibility available with this stochastic formulation. Instead, we've chosen a prior for this comparison based on Huber-Markov random fields, as demonstrated by Schultz and Stevenson [Schultz 1996]. This prior models piecewise smooth data, or smooth regions separated by discontinuities. This penalizes edges less than a Gaussian prior and is intuitively reasonable. The specific formulation and equations of the MAP algorithm with its Huber-Markov prior are laid out below.

3.2.1 Technical Description

The algorithm described below (and used in the comparison) is a hybrid of several MAP algorithms. A framework by Hardie was most amenable to coding, but Hardie's Gaussian prior was discarded in favor of Schultz and Stevenson's Huber-Markov Random Field (HMRF) prior [Hardie 1997, Schultz 1996]. The hybrid algorithm required an additional modifications, which is noted at the relevant point below.

To reiterate, a MAP algorithm chooses the $\hat{z}$ that maximizes the distribuition

$$\hat{z} = \arg\max_{z} \Pr\{z \mid y\} = \arg\max_{z} \left(\ln \Pr\{y \mid z\} + \ln \Pr\{z\}\right) . \tag{11}$$

Hardie shows that for Gaussian noise and a Gaussian prior, the above equation is equivalent to

$$\hat{z} = \arg\min_{z} \left[\frac{1}{2\sigma_n^2} (y - Wz)^T (y - Wz) + \frac{1}{2} z^T C_z^{-1} z \right] , \tag{12}$$

where σ_n^2 is the variance of the additive noise, and $\boldsymbol{C}_z$ is the covariance matrix of z [Hardie 1997]. We keep the first term of the sum (which is a likelihood term), but discard the second term (which relates to the Gaussian prior) and replace it with a Huber-Markov prior.

The Huber-Markov prior used by Schultz and Stevenson penalizes edges less harshly than the Gaussian prior does. The probability density of z is defined as

$$\Pr\{z\} = \frac{1}{Z}\exp\left\{\frac{-1}{2\beta}\sum_i \rho_\alpha\left((B_1 z)_i\right) + \rho_\alpha\left((B_2 z)_i\right) + \rho_\alpha\left((B_3 z)_i\right) + \rho_\alpha\left((B_4 z)_i\right)\right\} , \tag{13}$$

where Z is a normalizing constant, β is a parameter, and $()_i$ is the ith element of a column vector. The matrices $\boldsymbol{B_1}$, $\boldsymbol{B_2}$, $\boldsymbol{B_3}$, and $\boldsymbol{B_4}$ are defined so that $\boldsymbol{B_n z}$ has low values in smooth areas and high values at edges. Specifically, $\boldsymbol{B_1}$ is defined so that row d of $\boldsymbol{B_1 z}$ is equal to $z_e - 2z_d + z_f$, where z_e is the pixel to the left of z_d and z_f is the pixel to its right. $\boldsymbol{B_2}$ is defined so that row d of $\boldsymbol{B_2 z}$ is equal to $\frac{1}{2}\, z_e - z_d + \frac{1}{2}\, z_f$, where z_e is the pixel to the lower left of z_d and z_f is to its upper right. $\boldsymbol{B_3}$ is defined so that row d of $\boldsymbol{B_3 z}$ is equal to $z_e - 2z_d + z_f$, where z_e is the pixel above z_d and z_f is the pixel below it. Finally, $\boldsymbol{B_4}$ is defined so that row d of $\boldsymbol{B_4 z}$ is equal to $\frac{1}{2}\, z_e - z_d + \frac{1}{2}\, z_f$, where z_e is the pixel to the lower right of z_d and z_f is to its upper left.

Finally, the function $\rho_\alpha(x)$ is the Huber penalty function, which operates on the individual elements of the $\boldsymbol{B_n z}$ vectors:

$$\rho_\alpha(x) = \begin{cases} x^2, |x| < \alpha \\ 2\alpha|x| - \alpha^2, |x| \geq \alpha \end{cases} . \tag{14}$$

The function is pictured in Figure 7 along with a Gaussian penalty function.

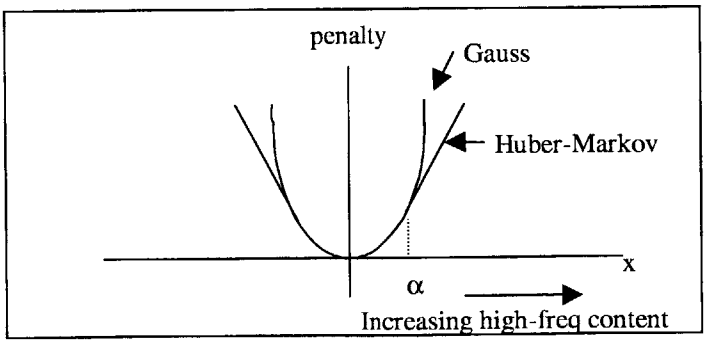


Figure 7 – Gauss and Huber-Markov penalties. Edge regions (high values of x) are penalized less severely by the HM function.

Schultz and Stevenson simplified the HMRF prior into a cost function similar to that of Hardie *et al.* Incorporating their results into Hardie's cost function (replacing Hardie's Gaussian prior) gives

$$\hat{z} = \arg\min_{z} \frac{1}{2}(y - Wz)^T (y - Wz)$$

$$+ \frac{\lambda}{2} \sum_i \rho_\alpha\left((B_1 z)_i\right) + \rho_\alpha\left((B_2 z)_i\right) + \rho_\alpha\left((B_3 z)_i\right) + \rho_\alpha\left((B_4 z)_i\right). \quad (15)$$

This equation is similar to the RMMSE equation from the previous section, and is easily minimized with a similar iterative algorithm [Hardie 1997]. The value for λ varies with the amount of noise, but is typically ~ 0.05-0.1. This algorithm minimizes the cost function by:

- Calculating the gradient
- Choosing a stepsize ε to move in the negative gradient direction
- Updating the estimate $\hat{z}$ to the new point and iterating

The gradient derivation follows the RMMSE derivation closely, except for the differences due to the smoothness term. The derivation is relegated to Appendix A since it would not provide additional understanding if listed here. After the gradient calculation, Hardie *et al*'s RMMSE method calculated a stepsize ε using the line minimization method. Because of the mathematical properties of their function, a closed form analytic solution for ε could be produced. In this MAP algorithm, on the other hand, a more complex function is being minimized, and a closed form expression for ε is more difficult to derive. For this reason ε must be calculated in some other manner.

At this point the implementation diverges from both published sources. With the gradient as given above, a step size ε is chosen by the Armijo rule. The Armijo rule is an iterative step size selection algorithm with good theoretical convergence properties, and Appendix A summarizes the inclusion of the Armijo rule. The stepsize ε produced by the Armijo rule guarantees that $\mathbf{C(z^{k+1}) < C(z^k)}$, and the entire process is iterated until the cost function converges to a minimum. The result is the MAP estimate of z. The Matlab code for this algorithm can be found in Appendix B.

3.3 Projection Onto Convex Sets

Projection onto convex sets (POCS) algorithms differ substantially from other super resolution algorithms (like MMSE & MAP) in both their formulation and their operation. Instead of minimizing or maximizing certain *quantities*, POCS algorithms produce an image that has a specific set of *qualities*. These qualities are chosen by the user and take the form of convex sets (a set is *convex* if $\alpha x + (1 - \alpha)y \in C$ for $\forall x,y \in C$ and $\forall \alpha \in [0,1]$). The POCS theorem guarantees that the result of the algorithm is a data set that satisfies every constraint set. POCS algorithms are easily extendible, and additions, omissions, or changes to the constraint sets require only localized rederivations. Unfortunately, POCS is intrinsically highly iterative, is hard to optimize for speed, and runs more slowly than many other algorithms.

The theory of convex projections forms the theoretical basis for POCS algorithms. Suppose it is known that a solution vector z has certain properties, and each property can be stated as a convex set. If we have m such constraints, then z is in the intersection of all the convex sets, or

$$z \in \bigcap_{i=1}^{m} C_i \quad . \tag{16}$$

Next, we define a projection operator $\boldsymbol{P_i}$ for each set where the result of applying $\boldsymbol{P_i}$ to any z results in $\boldsymbol{P_i z} \in \boldsymbol{C_i}$. The theory of POCS states that for any choice of z^0, the iteration $z^{k+1} = \boldsymbol{P_m P_{m-1}} \ldots \boldsymbol{P_2 P_1 z^k}$ will cause z to converge to a point in $\bigcap \boldsymbol{C_i}$, thus satisfying every constraint [Stark 1989]. This intersection may have more than one element, so the solution may not be unique and is usually affected by the starting point, z^0.

POCS restoration requires that one's knowledge about an image is expressible using convex sets. Fortunately, this is true of most qualities of interest. For instance, we know that a true image distribution has nonnegative intensity. This can be expressed as the convex set

$$C_A = \{z : 0 \le z\} \ , \tag{17}$$

where z is a column vector and $z \ge 0$ means that every element of z is ≥ 0.

The projection P_A onto set C_A can be simply defined as:

$$P_A z = \begin{cases} 0 & , z_i < 0 \\ z_i & , else \end{cases} \quad . \tag{18}$$

This nonnegativity constraint is just one example of the many constraints available. Other constraints can relate to the observed low resolution pixels, the energy of the reconstruction, *a priori* knowledge of the scene, or other more creative data sets.

Since all POCS iterations operate in the same manner, differences are produced solely by varying the choice, number, and content of the convex sets. The specific choices made for this thesis are given below and fully specified. Note that the inclusion of additional constraints would produce different results, so the power of POCS lies in the ability to choose the best constraint sets.

3.3.1 Technical Description

The specific POCS algorithm analyzed in this thesis is similar to Stark and Oskoui's formulation [Stark 1989]. The choice of convex sets and their associated projections are listed here. Once all sets are defined, the algorithm operates simply by iterating $z^{k+1} = P_m P_{m-1} \ldots P_2 P_1 z^k$ until $z^{k+1} = z^k$. At this point, z satisfies every constraint, and is thus a viable solution.

The constraint sets used for this thesis are an amplitude constraint, an energy constraint, a finite support constraint, and the data point constraints from Stark and Oskoui [Stark 1989]. Stark and Oskoui also include a reference-image constraint, but we omit it since it relies heavily on specific *a priori* information. The initial guess, z^0, influences the solution when $\bigcap C_i$ has more than one member. Stark and Oskoui demonstrated better performance when z^0 had similar structure to the true high resolution image (as opposed to a constant initial image). For this reason, z^0 was synthesized as an interpolation of the low resolution data points.

The amplitude constraint was mentioned above, and is the most simple. We know that the intensity of any pixel can never be negative, so we define

$$C_A = \{z : 0 \le z\} \tag{19}$$

$$P_A z = \begin{Bmatrix} 0 & , z_i < 0 \\ z_i & , \ else \end{Bmatrix} . \tag{20}$$

The energy constraint guarantees that the high resolution result does not have more energy than the low resolution actual data. This constraint is defined as

$$C_E = \left\{ z : \|z\|^2 \le E \right\} \tag{21}$$

$$P_E z = \begin{Bmatrix} z & if \|z\|^2 \le E \\ (E/\|z\|^2)z & if \|z\|^2 > E \end{Bmatrix} , \tag{22}$$

where E is the maximum energy allowed for the reconstructed image and $\|\mathbf{x}\|^2 = \sum_i x_i^2$. We typically define E as the energy in a low resolution frame.

The finite support constraint guarantees that the reconstructed image is bounded in space, and is defined as

$$C_S = \{z : z_i = 0 \ for \ i \notin A\} \tag{23}$$

$$P_S z = \begin{Bmatrix} z_i & if \ i \in A \\ 0 & else \end{Bmatrix} , \tag{24}$$

where A is a finite region of space, typically the region for which we have data points.

Finally, the observed pixel values provide the most constraints, and each pixel is treated as a separate constraint with a separate convex set and projection. We define the j^{th} data point's constraint as

$$C_j = \left\{ z : \left| y_j - (Wz)_j \right| < \delta \right\} , \tag{25}$$

where y_j is the j^{th} data point of the data vector y and $(Wz)_j$ is the corresponding value of the vector that results from applying the imaging model (W) to the high resolution estimate z. This says simply that if we use our high resolution estimate to produce a low resolution simulated data set, then our simulated data must conform to the observed data within some margin δ. The associated projection is defined as

$$P_j z = \begin{cases} z & , \quad \left| (Wz)_j - y_j \right| \le \delta \\ z + W^T \left(\dfrac{k_j}{(W^T W)_{(j,j)}} \right) & , \quad \left| (Wz)_j - y_j \right| > \delta \end{cases} , \tag{26}$$

where k_j is the vector with length equal to length(y) and with each element having value zero except for the j^{th} row, which has value $(y_j - (Wz)_j)$.

Note that Stark and Oskoui use $\delta = 0$. The data presented in Chapter 4 was created by setting δ to 3 times the standard deviation of the additive noise, or $\delta = 0.01$ in the case of no additive noise. This approach was taken from [Sezan Jan1990].

3.4 Benchmark Methods

The three super resolution algorithms in this thesis are analyzed and compared to some benchmark methods. The benchmark methods include a single frame bicubic interpolation, an inverse filter with interpolation, a multiple-frame interpolation, and a multiple-frame interpolation followed by inverse filtering. A brief summary of the benchmark methods is given here, and the Matlab code for their implementation is found in Appendix B.

The bicubic interpolation is implemented with Matlab's built-in two-dimensional interpolation operator (interp2). Its operation is straightforward, and details can be obtained from the Matlab software package or an image processing text such as [Lim 1990].

To process single frames of data, the inverse filter with interpolation employs a two step process which is shown in Figure 8. The initial step filters the data with a constrained inverse filter, and the second step interpolates the result. The filter in step one is implemented via Fourier transform. The SRF is transformed and inverted, with the gain of the inverse filter constrained to reduce spurious noise amplification (Gain < 10). The amplitude constraint is implemented so that phase information in the inverse filter is preserved. With the filter thus created, one frame of data is Fourier transformed, multiplied by the constrained inverse filter, and inverse transformed. The result is a new spatial-domain image, which is then interpolated up to the desired rate using the bicubic interpolator discussed above.

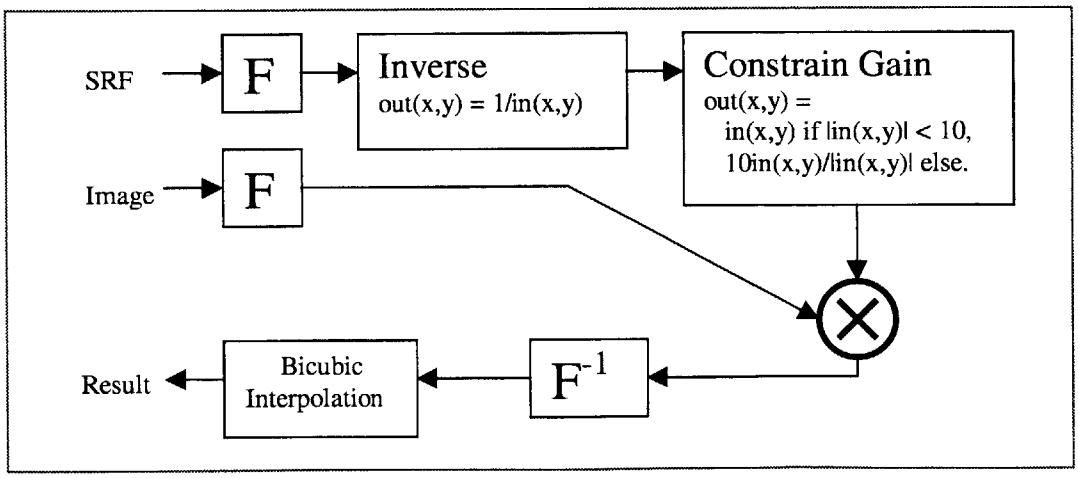


Figure 8 – Scheme for single-frame inverse filtering and interpolating. 'F' represents the two-dimensional Fourier Transform.

A multiple frame interpolator serves as another benchmark. Two multi-frame interpolators are actually used: one for uniform offsets and another for non-uniform. For uniform offsets, a high resolution grid is filled with sample values from all low resolution frames, and any empty grid points are filled by a bicubic interpolation. For non-uniform offsets, each high resolution grid point is determined by a weighted average of the three nearest low resolution pixels. For this scheme, the weights are inversely proportional to distance.

The final benchmark is created by passing the output of the multi-frame interpolator through an inverse filter. The inverse filter is implemented with an algorithm designed by Irani and Peleg [Irani 1991]. This algorithm is not linear; it was chosen because it amplifies noise less than a linear constrained inverse filter, and produces slightly better results. Again, the Matlab code for all methods is found in Appendix B.

Chapter 4 - Testing & Results

This chapter describes the empirical algorithm comparison and its results. Several test images are chosen and degraded in a variety of ways. The degradations are known by the algorithms, both super resolution and benchmark, which operate to produce estimates of the high resolution truth. The reconstructed high resolution images are compared to the high resolution truth with several mathematical metrics. In the Results section, the relevant results are displayed, showing the superior performance of SR algorithms over the benchmarks and the limited differences between SR algorithms.

The test situations were carefully chosen to answer the types of questions posed in the introduction about the performance and application of SR techniques. Specifically, the tests were designed to answer these questions:

- Are super resolution algorithms superior to linear filters and interpolations?
- What are the performance differences among super resolution algorithms?
- Is a single high-SNR image better than several offset low-SNR images?
- How does restoration relate to the number of frames and the uniformity of the offsets?
- What is the optimal ratio of aperture size to pixel size?

This chapter continues with a description of the testbed, metrics, and results. The Testbed Description discusses the image selection, the degradation process, and the parameters. The Comparison Metrics section presents and justifies the metrics used for algorithm comparison. The Results section displays the results for each test situation and makes some conclusions regarding the performance of the tested algorithms and how the data answers the questions posed above.

4.1 Testbed Description

This section describes the testbed and its parameters. The testing procedure is straightforward – determine high resolution "truth" images and the parameters to be varied, then for each image and parameter choice:

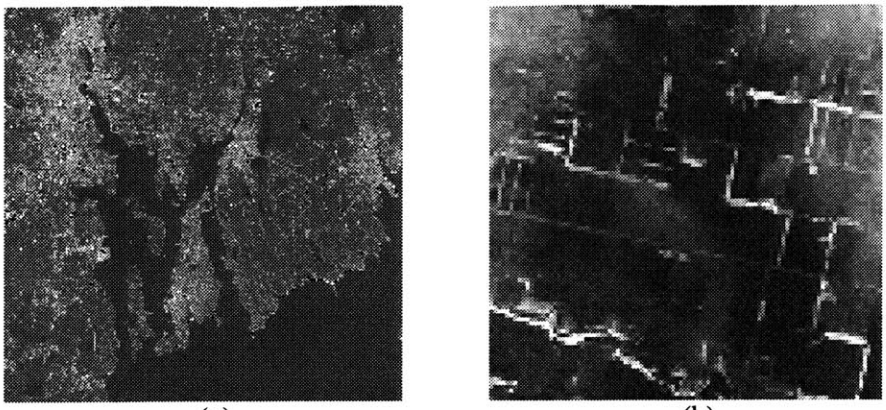
- Degrade the high resolution image to produce a low resolution data set
- Input the low resolution data to all algorithms
- Compare the high resolution outputs to the high resolution truth images.

The parameters, which are discussed below, are summarized here. Three high resolution truth images are used. One parameter is the number of low resolution (LR) frames available; this test uses 1, 2, 4, & 8 frames. Related to the number of frames are the offsets between frames; for each number of frames there are two different offset schemes. Another parameter is the amount of additive noise; we use four different Signal-to-Noise Ratios (SNRs), which are defined in Section 4.1.3. Finally, the Spatial Response Function (SRF) could be varied, but we've chosen a single SRF to limit the already large number of parameters. In general, the parameters were chosen with reference to the GOES8 satellite. The GOES8 was chosen to represent space-based imagers because Lincoln Laboratory has extensive GOES data.

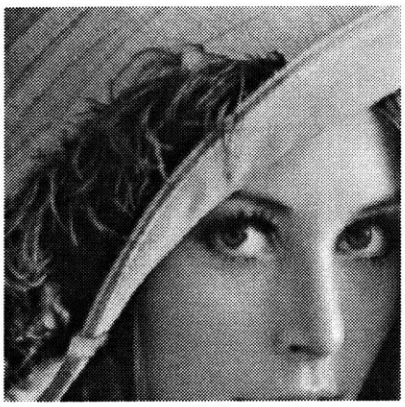
The tests were carried out in Matlab on a 200 MHz processor. The run times of the algorithms are not recorded because the implementations were not chosen for speed, were not exhaustively optimized, and because Matlab's lethargy in certain iterative computations would skew the results. Each parameter in the tests is discussed below, and all are followed by a description of the degradation process and the complete testbed specification.

4.1.1 Image Selection

Three high resolution truth images are used for testing. The three images are shown in Figure 9: a Landsat image of Rhode Island, another Landsat image of South Dakota, and Lena.



(a) (b)



(c)

Figure 9 – Three high resolution images used in algorithm tests. Image (a) is a Landsat image of Rhode Island, (b) is a Landsat image of South Dakota, and (c) is Lena.

The images were chosen for the variety of features they display. The Landsat images offer complex scenes that are typical for space-based imagers. The Rhode Island image is taken with clear skies; land, cities, and ocean provide edges and small features. The South Dakota image is taken with cloud cover, which provides high contrast boundaries between clouds and earth. Finally, the Lena image is regularly used in image processing literature and provides a simpler scene that is easy for humans to interpret.

4.1.2 Frames and Offsets

The performance of super resolution algorithms should improve with the number of frames available and should depend upon the offsets between frames. For this reason, we examine one, two, four, and eight frame reconstruction and we vary the offsets in the multi-frame cases. The particular offsets used in the tests are shown in Figure 10 using the notation developed in Chapter 2, Figure 4.

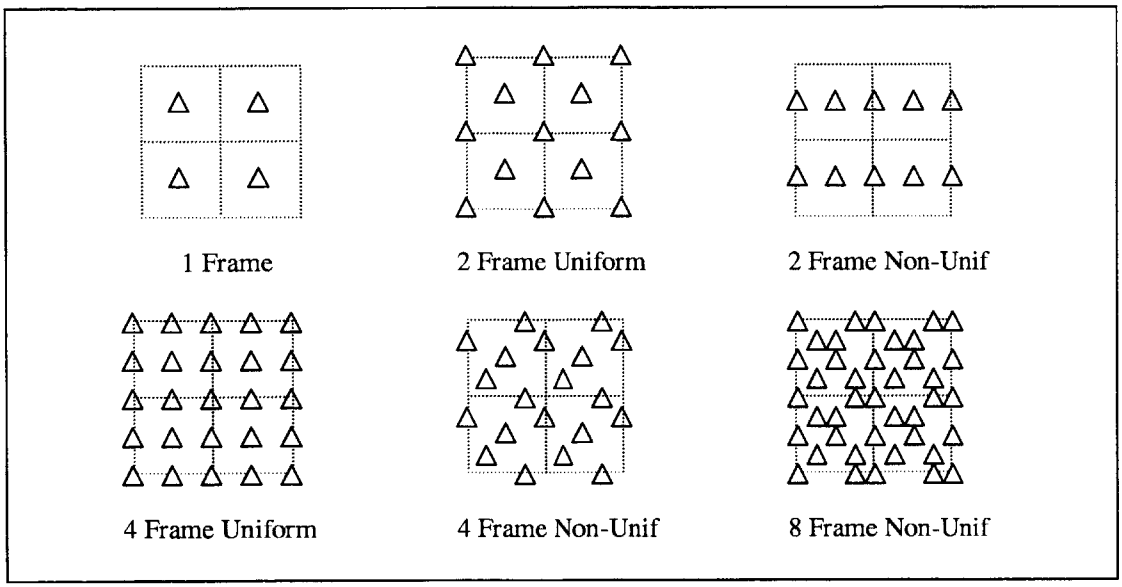


Figure 10 – Choice of offsets. For the various test situations, the offsets between low resolution frames are shown above. The triangles represent the centers of low resolution pixels, and the dotted lines show the size of a low resolution pixel for perspective (see Chapter 2, Figure 4 for more details).

For the single frame case, there is obviously no variety possible. For the two frame case a diagonal offset was used to provide a uniform sampling grid and a purely horizontal offset was used to produce a non-uniform data set, which is similar to the data set from a GOES8 satellite. For the four frame case there is both a uniform grid and a grid with the three additional frames positioned randomly. Finally, for the eight frame case a random choice of offsets was made.

The numerical values of the offsets are given in Table 1. For each situation, the offsets of all frames are given relative to the first frame (defined as offset (vertical = 0, horizontal = 0)). The offsets are measured in units of low resolution pixels.

2 Frame Uniform	2 Frame NonUn	4 Frame Uniform	4 Frame NonUn	8 Frame NonUn
(0, 0) (0.5, 0.5)	(0, 0) (0, 0.5)	(0, 0) (0, 0.5) (0.5, 0) (0.5, 0.5)	(0, 0) (0.25, 0.75) (0.5, 0.25) (0.75, 0.5)	(0,0) (0, 0.5) (0.25, 0.25) (0.25, 0.75) (0.5, 0.25) (0.5, 0.5) (0.75, 0) (0.75, 0.75)

Table 1 – Offsets for multi-frame test data.

4.1.3 Signal-to-Noise Ratios

Another important restoration parameter is the signal to noise ratio (SNR). We would like to determine if super resolution algorithms perform better or worse than traditional methods as the noise increases. For our tests we assume additive Gaussian noise, a reasonable assumption based on typical noise sources and the aggregate result of many independent sources.

There are several ways to define SNR, and the following method used by Lincoln Laboratory is adopted by this thesis: the SNR is defined as the peak signal value divided by the standard deviation of the noise. For the GOES8 imager, a modern weather satellite, the SNR is approximately 300.

For this testbed, four noise levels are used: SNR = infinity, 300, 150, & 75. The infinity case, corresponding to no noise, provides a baseline for systems with extremely low noise. The SNR = 300 corresponds to a GOES-like value. The SNRs = 150, 75 are chosen to study the effects of additional noise on the reconstruction.

4.1.4 Spatial Response Functions

A final parameter to be varied could be the Spatial Response Function (SRF). Changes in the SRF, which blurs the image, can strongly affect the resultant restorations. Also, the relationship between the SRF and the spatial sampling rate is important. Varying the SRF may produce interesting results, but in order to limit the extent of an already large test matrix, we have used only a single SRF. The SRF used was calculated during calibration of the GOES8 satellite, and is shown in Figure 11. Note that this SRF is asymmetric.

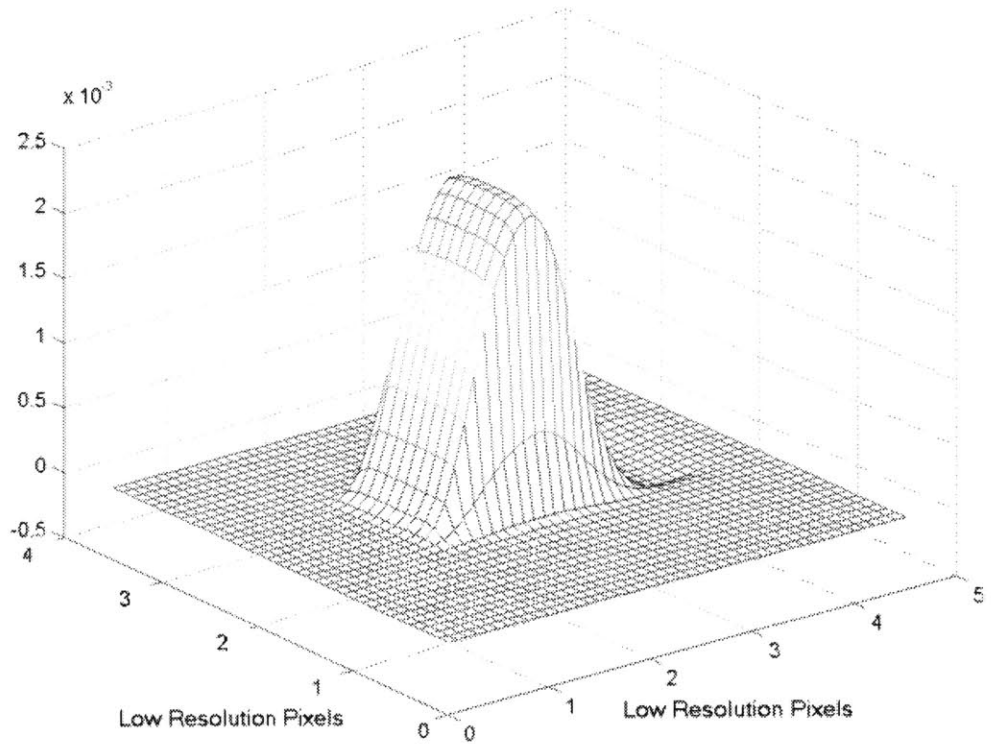


Figure 11 – Spatial Response Function used in tests. This SRF was characterized for a GOES weather satellite imager.

4.1.5 Degradation Process

The parameter choices are discussed above; each choice of parameters produces a set of low resolution images to be processed. We now discuss the specifics of how the parameters are combined to form the simulated data sets.

For each choice of parameters, the following process produces the final low resolution data. The high resolution "truth" image is convolved with the SRF to produce a blurred high resolution image. This blurred image is then downsampled to the desired low resolution of a single frame. The downsampling process is repeated for the number of frames chosen, with the image shifted with each downsample to correspond to the chosen offsets. At this point, a number of low resolution data frames exist which correspond to blurred, downsampled, offset images. Finally Gaussian noise is added in a quantity determined by the chosen SNR and the peak value of the blurred and downsampled images. The entire process is shown graphically in Figure 12.

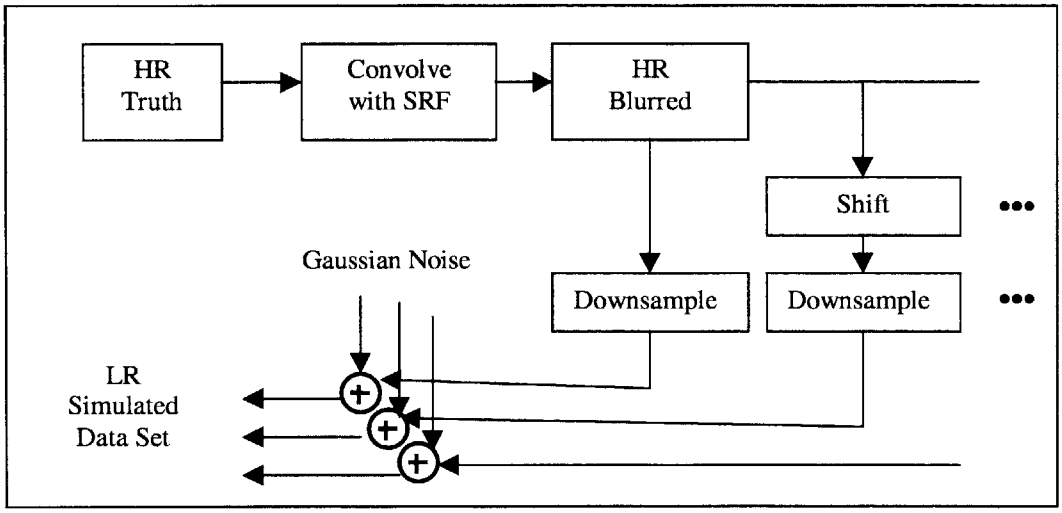


Figure 12 – Schematic of degradation process to produce low resolution data sets.

Each low resolution data set produced in this manner is presented to the restoration algorithms. The complete testbed specifications are given below.

4.1.6 Complete Testbed Specification

The testbed is completely specified in this section: all parameters are listed, all algorithms, and all the information presented to each algorithm. This section includes a discussion of why certain facts (e.g. offsets) are considered known, and need not be estimated by the algorithms.

All the imaging parameters that vary are summarized in Table 2. A low resolution data set is produced, as directed above, for each different set of parameters.

	SNR	1 Frame	2 Frame Uniform	2 Frame NonUnif	4 Frame Uniform	4 Frame NonUnif	8 Frame NonUnif
Image 1	infinity						
	300						
	150						
	75						
Image 2	infinity						
	300						
	150						
	75						
Image 3	infinity						
	300						
	150						
	75						

Table 2 – Summary of all testing parameters. For each element in the table, all algorithms operate on the test data and all metrics are calculated on each result.

For each data set, all three super resolution algorithms operate on the data. The single frame bicubic interpolation and inverse filter algorithms also operate on every data set (for multi-frame sets, they operate on the first image of the set). For multi-frame sets, the appropriate multi-frame interpolation is also run. For every restoration, the algorithms are operated to increase resolution by a factor of two in each direction (a single factor was chosen to limit the size of an already huge test matrix). This choice is arbitrary, but is not as restrictive as it appears: since the number of frames varies, it is possible to extrapolate to more general results. For instance, we could estimate that performance with 8 data frames and a resolution increase of 4X in each direction should be similar to the performance observed with 2 data frames and an increase of 2X in each direction.

Finally, each algorithm is given accurate estimates of the Spatial Response Function (SRF), the multi-frame offsets, and the noise level. Many procedures exist for determining these quantities [Shekarforoush 1998, Kaltenbacher 1996], which are generally over-specified by the data. In addition, specific information is often known *a priori*, such as calibration results and SRF estimates. For these reasons, it is reasonable to assume that this data can be calculated accurately and passed to the processing algorithms. In our tests, no error was introduced into the data, but doing so would be more realistic and is discussed in the Future Work section of the Conclusion.

For each low resolution data set, then, each algorithm produces a single high resolution reconstruction. The comparison of these outputs is discussed in the next section, which in turn is followed by the results.

4.2 Comparison Metrics

The comparison of images is difficult: no single metric has been devised as an authoritative measure of image accuracy. This is particularly true within the super resolution literature, where many presentations lack mathematical comparisons altogether. Without any performance metrics, super resolution algorithms cannot be compared. For this reason, we have compiled several metrics which, when used together, provide acceptable results. These metrics are not a panacea to the comparison problem, but for our tests have provided a reasonable mathematical basis.

The two mathematical metrics used are a normalized mean squared error (MSE) and a normalized Laplacian mean squared error (LMSE). The metrics require both the existence of a high resolution truth and a certain relationship between sampling rates. The fact that the high resolution truth is known is assured by the testbed setup (discussed above). The required relationship between sampling rates is that both the truth and the reconstruction must have equivalent sampling rates. This could be accomplished by either decimating the truth image to the reconstruction rate, or interpolating the reconstruction up to the truth rate. Since the results are equivalent for perfect decimation and bandlimited interpolation, the decimation is chosen for its lower computational complexity.

The mean squared error between two signals is a common metric, which is mathematically defined as:

$$MSE = \sum_x \sum_y [z(x,y) - \hat{z}(x,y)]^2 \quad . \tag{27}$$

We normalize the MSE by dividing the Equation 36 by the energy of the truth image, which is simply the sum of the squared values of every pixel in the image.

Also, the LMSE is defined as:

$$LMSE = \sum_x \sum_y [g(x,y) - \hat{g}(x,y)]^2 \quad , \tag{28}$$

where $g(x,y)$ is defined as

$$g(x,y) = z(x+1,y) + z(x-1,y) + z(x,y+1) + z(x,y-1) - 4z(x,y) \quad . \tag{29}$$

The Laplacian MSE is taken from Pratt [Pratt 1978], and it weighs edge regions more heavily than the pure MSE does. It is a good measure of the effective low-pass filtering, since it drops off more quickly than the MSE measure as low-pass filtering increases. Just as with the other two metrics, we normalize the LMSE, in this case by dividing by the energy of $g(x,y)$.

A few additional steps help create a more accurate comparison. First, the border regions of images can be corrupted by processing. To account for this, the metrics are operated on both the full images and the images with a border region removed. Second, the shifts between low resolution images could cause uncertainty in the alignment of the high resolution reconstruction (as compared to the original). This problem is eliminated by aligning the images (using a cross correlation) before running the metrics. See the Matlab code in Appendix B for details of the comparison process.

4.3 Results

The tests outlined above produced numerous results. Both metrics were computed for the output of every algorithm in every test situation (see Table 2). The complete data set is contained in Appendix C, and below are plots and sample images that display the results in a graphical format.

In the tables below, POCS data for the nonuniform 4- and 8-frame cases are absent. This is not due to any theoretical limits of POCS; it is only the result of an implementation problem and the author's time constraints. Also, the plotted errors are from the metrics operating on the images with the borders removed. The results are similar for the border regions included, as can be seen in the data in Appendix C.

4.3.1 Aggregate Results: SR algorithms and benchmarks in all situations

The overall behavior of the three chosen super resolution (SR) algorithms is impressive, especially as the number of data frames increases. Figure 13 displays two graphs, one for each metric. The plotted error values are the sum of the errors from the three test images. They are shown for all combinations of SNR, number of frames, and offsets. The POCS data for the 4- and 8-frame nonuniform situations was not computed, as discussed in the previous section.

These results clearly show the following conclusions:

- The differences between SR algorithms and benchmarks are miniscule in the single frame case, but expand dramatically as more data frames are available.
- The three SR algorithms all exhibit similar performance.
- The POCS algorithm performs better in the case of no noise and worse with noise. This matches expectations, since the POCS implementation used here has no smoothness constraint; with no noise, it is not over-smoothing the data, but with noise, it is more susceptible to errors.

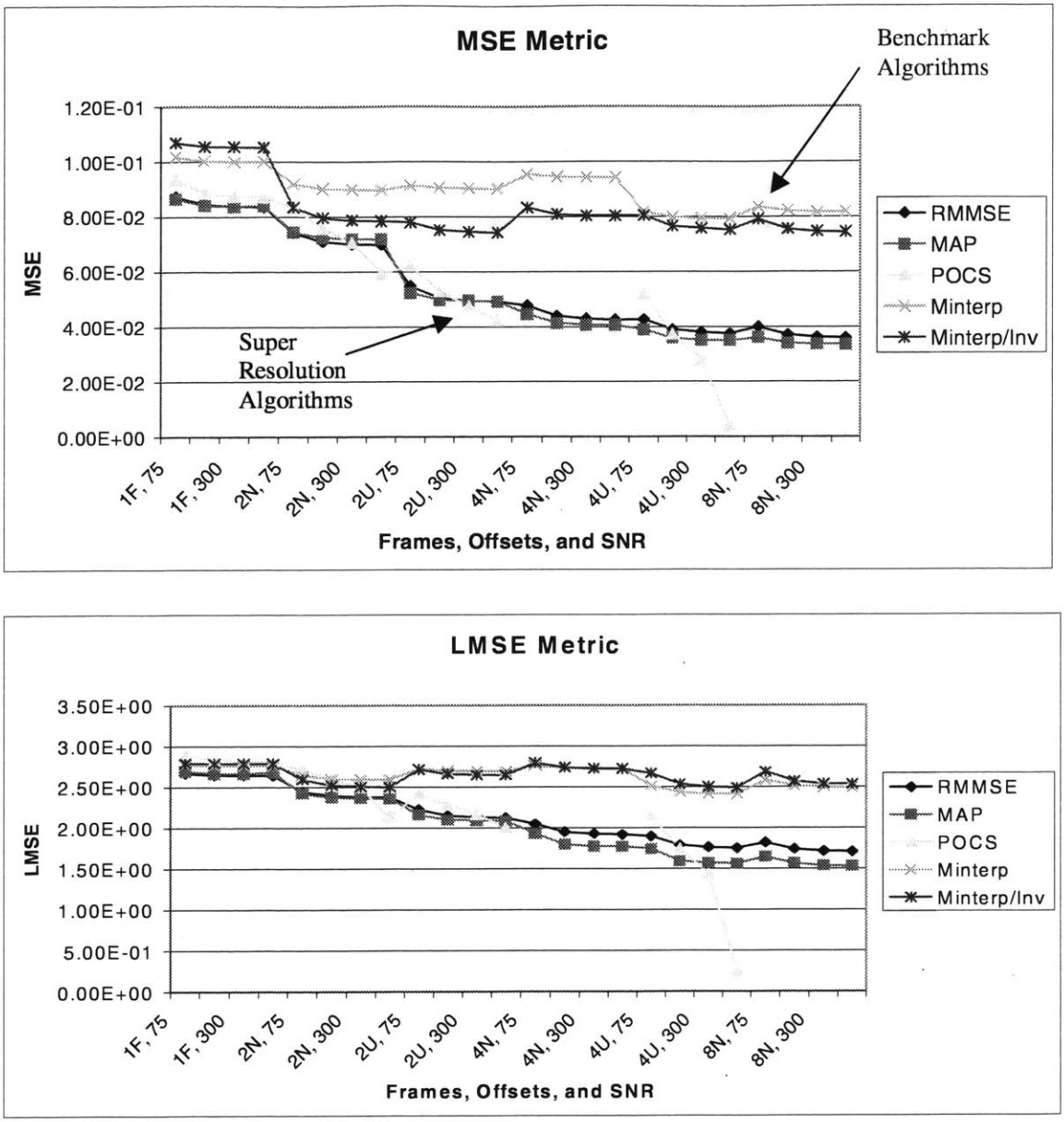


Figure 13 – Aggregate performance. The two error metrics are plotted verses the test situation. From left to right, the SNRs are 75, 150, 300, and infinity for each frame/offset combination.

More specific comparisons will come later along with more concise, revised plots. Nevertheless, the aggregate behavior displayed by this full data set is impressive. Super resolution algorithms perform substantially better than multi-frame interpolations and inverse filters regadless of the uniformity of the offsets. To confirm these metrics subjectively, a few sample images are shown

below in Figure 14. Besides the general impression of sharpness, note the smaller features which can be found in the multi-frame super resolution reconstructions but not in the benchmarks.

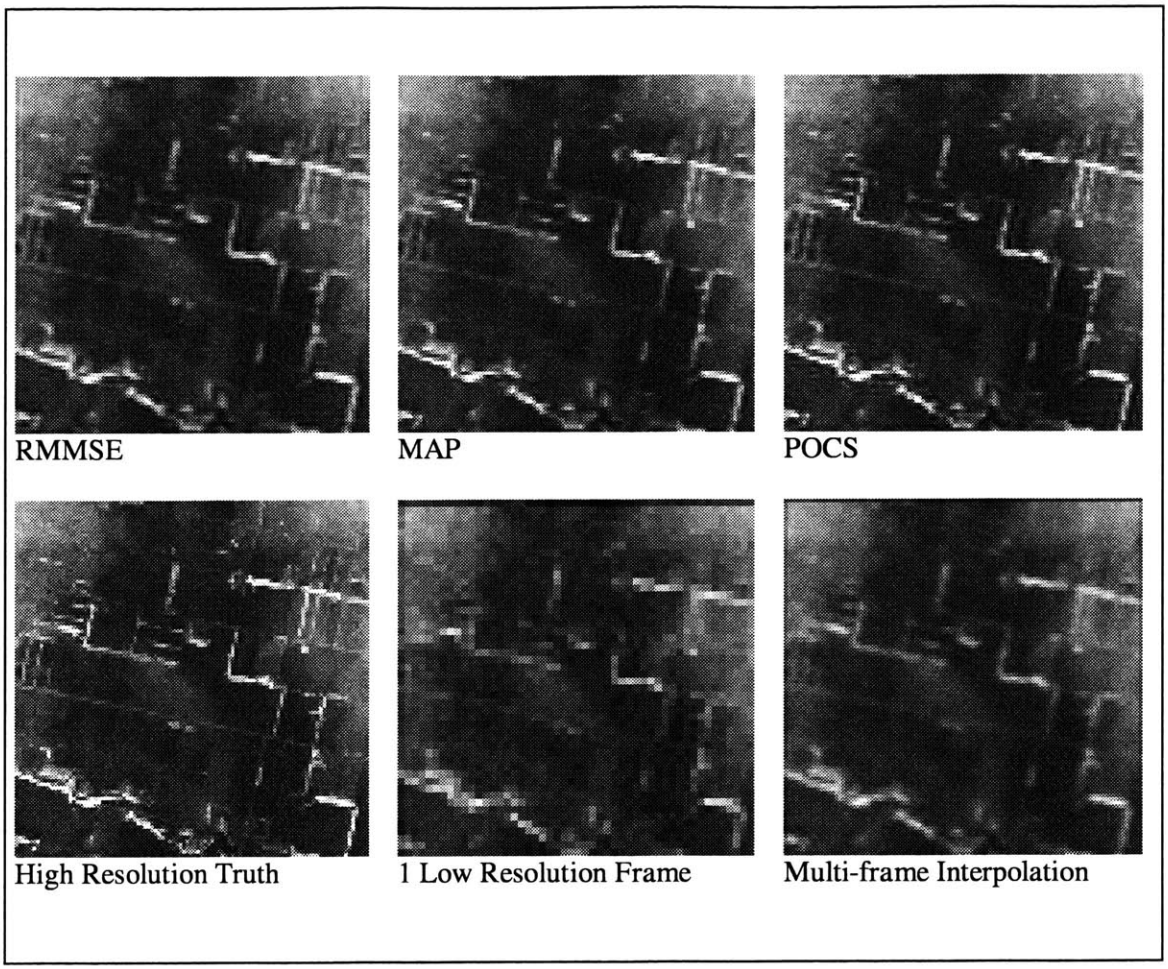


Figure 14 – Selected images for 4 Frame, uniform offsets, SNR 300, South Dakota image. Note the features that can be seen in the super resolution images but not the degraded or interpolated images.

4.3.2 Noise Effects

Noise affects all reconstructions and presents interesting questions. The most obvious question is for all other factors constant, how does more noise degrade the reconstruction? The answer, which can be seen in the Figure 13, is that the effect is small. This is because most algorithms (benchmark *and* super resolution) perform smoothing, which masks noise (as well as the signal).

The algorithm without a smoothing mechanism (POCS) performed best in the no noise case but was harmed most by noise increases.

An additional question about noise relates to the design of imaging systems. Suppose the frame rate of an imager could be increased at the expense of a lower SNR for each frame. If offsets could be introduced between frames, then capturing several low-SNR images may be superior to capturing a single high-SNR frame. The test data is arranged in Figure 15 to show how super resolution algorithms perform as the SNR is lowered and more frames are added.

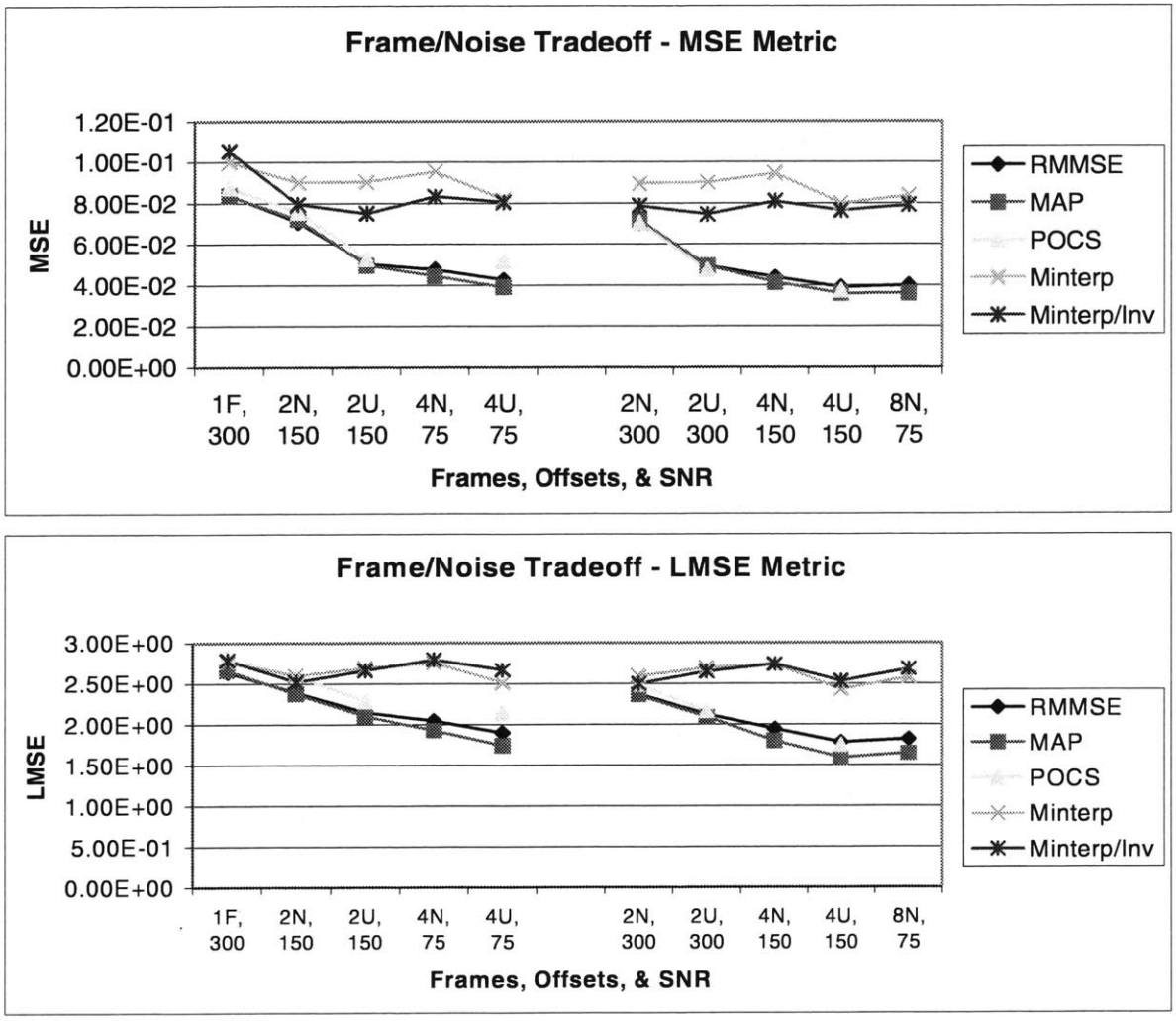


Figure 15 – Frame/Noise Tradeoff. As the number of frames doubles, the SNR is halved for this test. Note that for the super resolution algorithms, a situation with more low-SNR frames is preferable to one with fewer high-SNR frames.

The plots above show the definite trend that more frames are preferable to a high SNR. An interesting extension could experiment with a wider range of SNRs and frames to seek fundamental limits. Again, to show some true images, Figure 16 displays some samples.

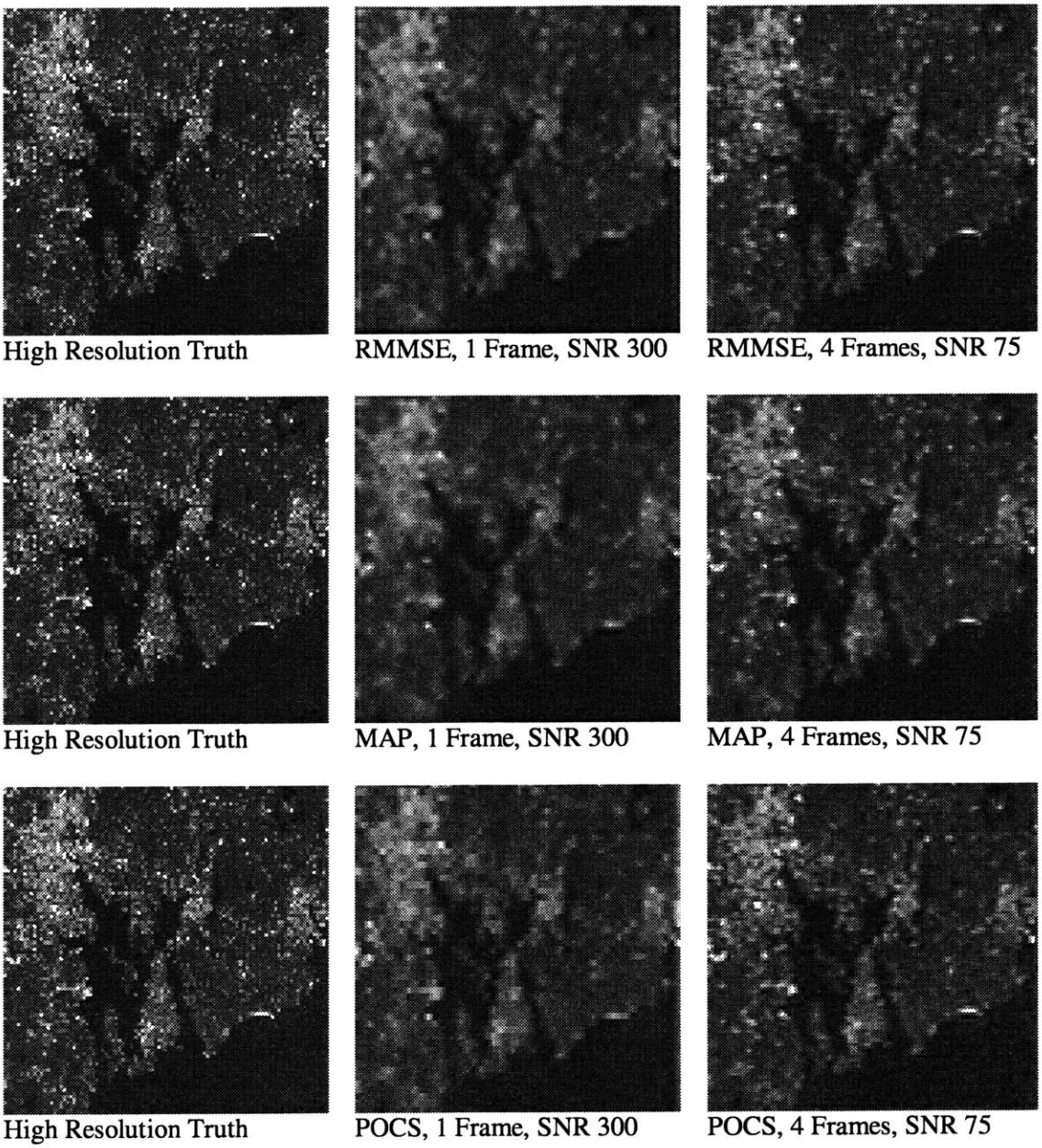


Figure 16 – Sample images for Frame/Noise tradeoff. A truth image is displayed along with the result of the three algorithms (RMMSE, MAP, & POCS) operating on a single frame with high SNR and multiple frames at a lower SNR.

4.3.3 Frames and Offsets

Figure 13 showed the general trend that as the number of frames increases, the accuracy increases for all multi-frame algorithms, and particularly super resolution algorithms. These relationships are examined more closely in Figure 17 below.

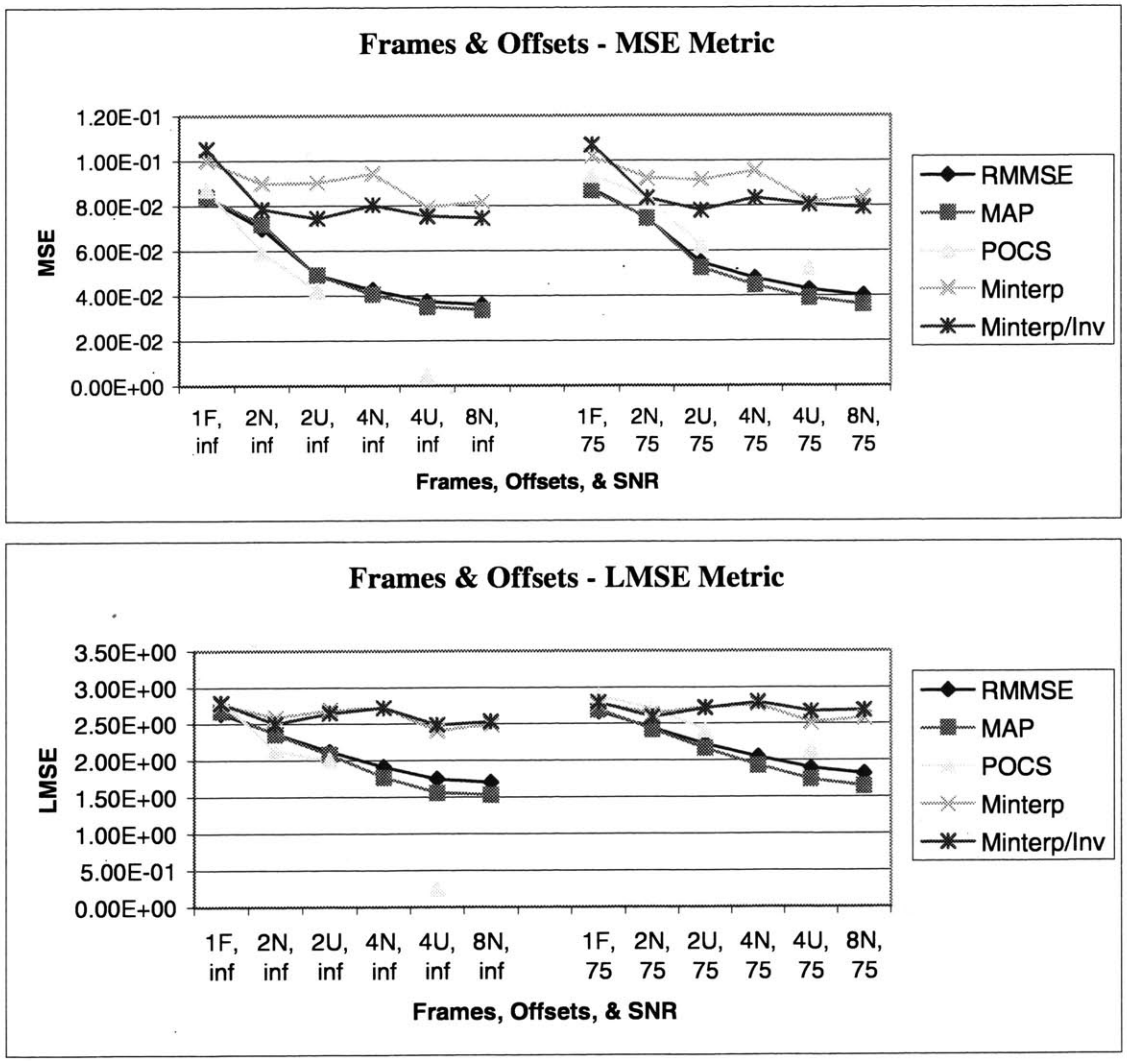


Figure 17 – Frames & Offsets Comparison. The metrics are plotted for each algorithm, and the horizontal axis is arranged with adjacent points having equivalent SNRs. This allows easier comparisons between different frame/offset pairs. The results for SNR = 300, 150 were omitted to make the plots more readable, as they are similar to the shown plots.

The graphs in Figure 17 clearly show the advantages of multiple frames of data. Super resolution algorithms are particularly adept at extracting information from additional frames, shown by the monotonically decreasing error. Note that the uniform cases are always more accurate than the random offsets, as expected. Nevertheless, capturing more frames is always preferable, even if the offsets are random. Note also that the relationship between the number of frames and the error is nonlinear and dependent on the SNR. In general, though, the error was approximately halved by going from one frame of data to eight.

Regarding the benchmarks, the multiple frame interpolation selectively followed by an inverse filter, there is only slight inconsistent improvement as the number of frames increases. All of the multiple frame interpolations exceed the accuracy of the single frame, but after that point, the performance is erratic. This is acceptable, because the offsets and the interpolation process can interact negatively; the interpolation, since it is not perfect and bandlimited, can introduce spurious data. This spurious data is filtered along with the true data, harming results. In super resolution algorithms, on the other hand, the filtering process and the offsets are treated concurrently, which eliminates the distortion introduced by a pure interpolation followed by a filter.

Sample images are displayed in Figure 18 to confirm the numeric findings. Notice the features that are present in the truth, gone in the degraded and single frame reconstructions, but present again in the higher frame-number reconstructions.

4.4.4 Image Differences

All the results above have summed the errors of the three test images. This is acceptable because the results are similar across images, so the summation allows a more concise view of the data without occluding information. Nevertheless, there are some small differences among algorithms for the various images that are useful to observe. Figure 19 displays error plots for the three images separately.

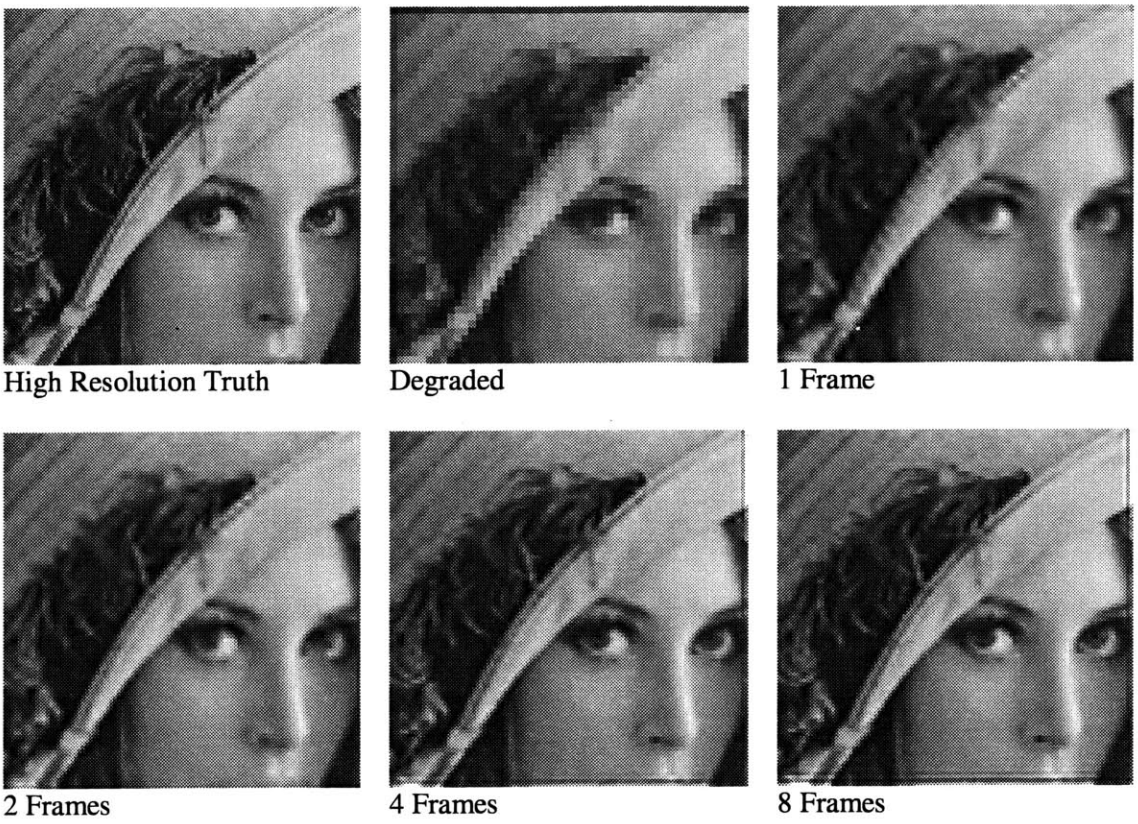


Figure 18 – Differences from number of frames. All of the above images were created by the MAP algorithm. The degraded images all had SNR = 150. The number of frames is shown below each image, and the offsets were random in the multi-frame cases. (Note the ringing introduced at the border of the image as the number of frames increases. This ringing occurs because the low resolution frames are offset from each other, and the information at the image boundaries is irregular).

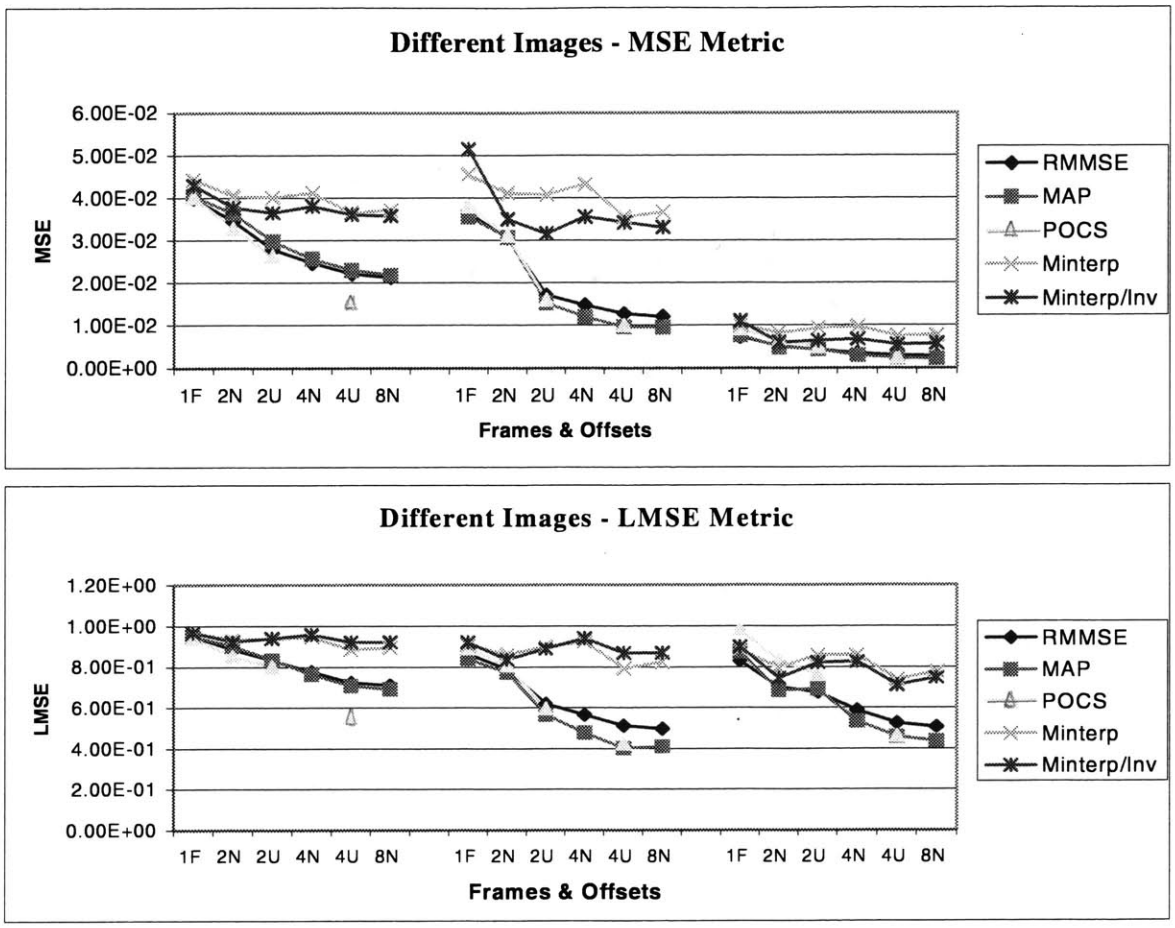
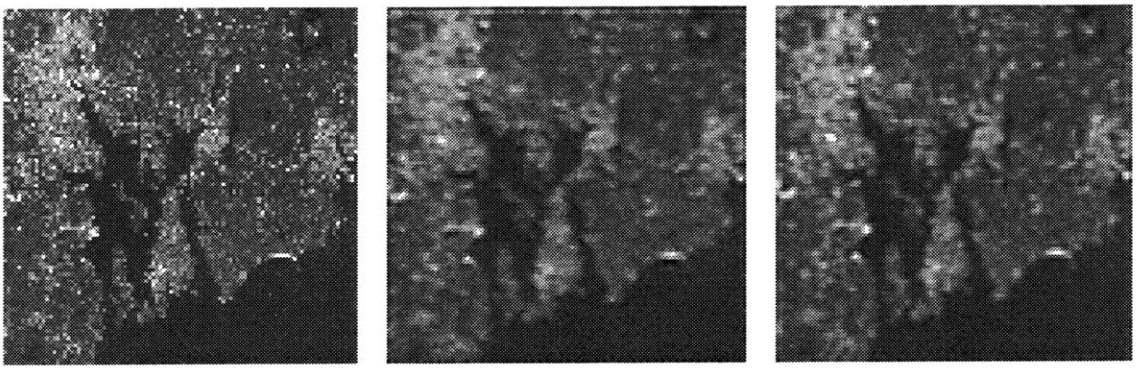


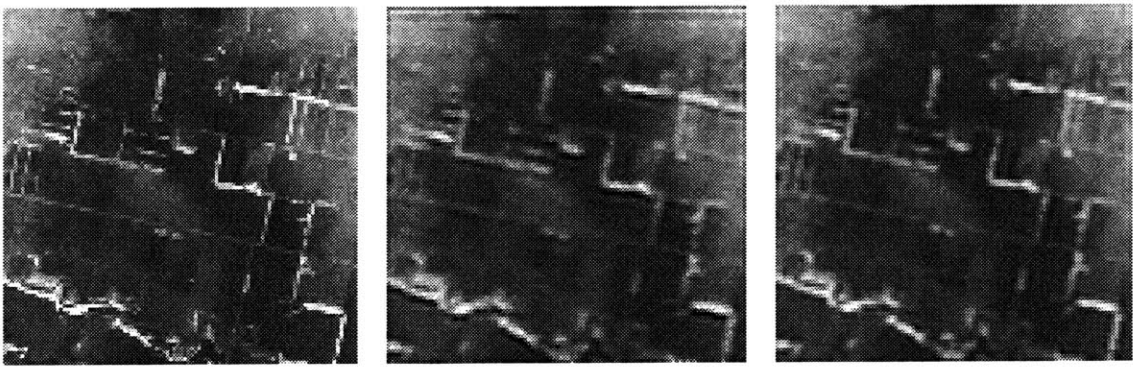
Figure 19 – Error for the three different test images. The data is shown for a single SNR = 300. Each plot has three separate regions: the first is Image 1, Rhode Island, the second is Image 2, South Dakota, and the third is Image 3, Lena.

The conclusions drawn from Figure 19 help develop and confirm intuitive notions of performance. For instance, the MSE metric has the highest error for Rhode Island and the lowest for Lena. This is expected, since the Lena image is the most simple, with large uniform regions, while Rhode Island is most complex, with substantial higher-frequency content throughout the image. We expect the low-frequency smooth regions to be reconstructed accurately, so Lena, and to a lesser extend South Dakota, will have their MSE lowered by the miniscule error in the smooth regions. In contrast, the Laplacian MSE metric has similar errors for all three image. This is because the LMSE weighs the high-frequency content of images; without their smooth regions to lower the average, Lena and North Dakota have similar error levels to Rhode Island. Finally, disregarding the absolute level of error, note that the three images are affected similarly by the algorithms. Some sample images are shown in Figure 20.

High Resolution Truth | Multi-frame Interp. & Inv. | RMMSE algorithm



High Resolution Truth | Multi-frame Interp. & Inv. | RMMSE algorithm



High Resolution Truth | Multi-frame Interp. & Inv. | RMMSE algorithm

Figure 20 – Super resolution algorithms affect the three test images similarly. For each image, the high resolution truth is shown along with the multiple frame interpolation and inverse filter benchmark and the RMMSE reconstruction. The data consisted of two frames with a uniform offset and a SNR = 300. The improvement over the benchmark is similar for all three images.

The results and commentary above have answered many questions about the performance of super resolution algorithms. This analysis has shown that these super resolution methods outperform benchmark interpolations and inverse filters when multiple frames of data are available at non-integer offsets. In all tested ranges of SNR, super resolution algorithms excelled, regardless of whether the offsets between frames were uniform or random (except for the POCS exceptions as noted). No major peculiarities exist in the data. In general, all three super resolution algorithms produced similar results; the exception was POCS, which lacking a smoothing operator, did well with no noise but suffered as noise was added.

The conclusion follows in Chapter 5 with a recapitulation of the questions, answers, and the supporting data. Chapter 5 also discusses extensions or improvements on this work that could be pursued.

Chapter 5 - Conclusion

This thesis has provided the first known comparison encompassing several super resolution and traditional algorithms. Three multi-frame algorithms (RMMSE, MAP, and POCS) were selected as a sampling of the super resolution field. These algorithms were compared to each other and to a set of benchmarks, which included inverse filtering and single and multiple frame interpolations. All algorithms were coded in Matlab and tested on a series of simulated data with varying frames, offsets, signal-to-noise ratios, and images.

In summary, the super resolution algorithms consistently outperformed the benchmark methods. When only a single frame of data was available the differences were small, but as more frames were added, the differences became dramatic. The uniformity of the offsets between frames had only a minor impact on results. Lower Signal-to-Noise Ratios (SNRs) led to less accurate reconstructions, but SNR was the parameter that affected the results least significantly. The overwhelming factor, as expected, was the number of frames available.

A more specific discussion of the results follows. The questions presented in the introduction are listed below and answered subsequently:

- Are super resolution algorithms superior to traditional linear techniques?
- Are there substantial differences among the various super resolution algorithms?
- How does noise affect performance? Are results more accurate if a few high-SNR frames are available or many low-SNR frames?
- How does the number of frames and offsets affect performance? Are random offsets as helpful as a uniform grid?
- What is the optimal ratio of aperture size to pixel dimensions?

This thesis has shown that super resolution algorithms are indeed superior to traditional restoration techniques. The data in Figure 13 clearly shows that super resolution (SR) algorithms reconstruct images with less error than the benchmarks mentioned above. The differences are small, as expected, when only a single frame of data is available. As more offset frames are

available, though, the SR algorithms excel. The mathematical metrics show this effect, and subjective image viewing confirms the variation in level of detail.

Though super resolution algorithms are superior to traditional methods, the performance differences *among* super resolution algorithms are small. Note how closely the error curves follow each other in Figures 13, 15, 17, and 19. This behavior is reasonable because all SR algorithms use the same information and follow the same general rule: to construct a viable high resolution image that would produce the observed samples. A contrast exists between POCS and the other two algorithms, and is due to the lack of a smoothness operator in this particular POCS implementation. Without a smoothing regularization, POCS should perform better in low noise and worse in high noise; this performance was observed (note that a POCS smoothness constraint could be defined, but it was not in order to introduce variety). It appears that the basic algorithm is not vital to super resolution, but the regularization and weighting terms are closely tied to performance in noise. In general, POCS was the algorithm most suitable to changes and expansions, and may be the best choice for continuing experiments. POCS requires a lengthy iterative implementation, however, and was the slowest of the tested algorithms.

It was found that noise affects reconstructions, but not nearly as much as other factors. Figure 15 illustrates that doubling the number of frames and halving the SNR produces consistently better results among SR algorithms. This suggests a future shift in imager design: imagers should take many offset frames, even at the expense of SNR, and let the processing software produce accurate composite images. Exceptions to this rule appear when the SR algorithm is susceptible to noise (notice the POCS curves, which flatten out more quickly). Thus a balance can be reached between the number of frames and the SNR, that balance being decided by the choice of algorithm.

Regarding the frames and offsets, the results provided some new information but mostly affirmed expectations. We expected the accuracy to increase as the number of frames increased, and this effect was observed. We also expected the accuracy to be higher for uniform than non-uniform offsets, and this phenomenon was likewise observed. We expected the incremental improvement to fall as more frames were used; Figure 17 shows that the improvement between 4 and 8 frames is small and the error curve begins to level off (additional experiments with higher numbers of frames could confirm this). Finally, it was found that more frames are preferable, even if the offsets are more random for the larger number of frames.

The question of optimal ratio of aperture size to pixel dimensions was not answered. The experiments were not completed due to time constraints. The question is compelling, though, and could be answered with future experiments as discussed below.

The overall results are consistent with the claims made in the super resolution literature. Super resolution algorithms do outperform interpolations and offer more accurate reconstructions in most cases.

5.1 Future Work

This thesis has produced interesting results, but many additional questions remain unanswered. One line of inquiry could extend this basic framework to more algorithms. Another possibility would follow the effect of more parameters, such as noise in the SRF characterization or offsets. An ambitious extension would be to devise mathematical comparison metrics that are widely accepted as accurate measures of image quality. Finally, ambitious research could attempt to unify the disparate super resolution work into a theoretically cohesive field. These four possible extensions are discussed below.

A straightforward extension of this thesis could be a similar testbed with additional algorithms. We have examined three super resolution algorithms (RMMSE, MAP, and POCS), and chosen a single implementation of each algorithm (see Chapter 3). The results have been compared with several simple metrics and some qualitative visual analysis. A useful comparison could encompass more algorithms and more implementations. There are countless other super resolution algorithms in existence, including different implementations of RMMSE, MAP, or POCS [Cheeseman 1994, Cohen 1998, Irani 1990, Sezan Jan1990]. Even for the implementations in this thesis, different smoothness parameters could be introduced. The POCS algorithm used in this thesis could be extended to work correctly with randomly offset data, also.

Another straightforward extension could vary different parameters. We have varied the offsets, SNRs, number of frames, and images; we have ignored different spatial response functions (SRFs), different factors of resolution increase, and noisy estimates of the SRF or offsets. In reality, each digital imager has a different SRF. The SRFs can vary widely, and are a main

contributor to degradation. An interesting comparison would gauge algorithm performance as the SRF became larger or smaller in spatial extent. The factor of resolution increase could also be varied. Finally, we have used correct SRFs and offsets, but in reality they can never be known exactly. The results of erroneous estimates could be explored to give more realistic information.

A more difficult task would be the development of new mathematical metrics for comparisons. After decades of image processing, few metrics exist for the comparison of images. The results of the metrics that do exist are considered suggestive rather than authoritative. A great benefit to image processing would be a metric or series of metrics that are universally accepted. To create such a metric is difficult, though, especially considering the disparate demand of image users: edge regions may be interesting to some while smooth areas are relevant to others. Nevertheless, advanced metrics could be devised, perhaps as the result of rigorous modeling of human vision and perception, which could surpass today's simple metrics.

Finally, super resolution research could be continued along theoretical lines towards a unification of the field. There is little cohesiveness or organization in the super resolution field. Many independent algorithms have been developed, but have not been compared either theoretically or empirically. This thesis has provided an empirical comparison of certain algorithms, and the results beg for more research and a better understanding of the field. Unifying research could:

- derive theoretical limits of performance
- compare algorithms theoretically
- unify the variety of techniques, including algorithms that utilize stereoscopic perspective changes, multiple independent motion between frames, and multispectral data.

The conclusion of this thesis is that super resolution should continue to be actively researched and analyzed. This thesis has illustrated by empirical comparison that super resolution algorithms are useful and powerful. It has also shown that the availability of super resolution processing may change future imaging system designs. As computational power continues to increase, super resolution algorithms will become even more useful and powerful. The results demonstrated in these initial experiments are encouraging.

Appendix A – Derivations

This Appendix contains certain derivations as referenced in the text:

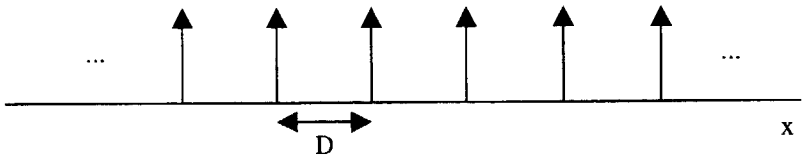
1) Derivation of the spatial sampling rate
2) Derivation of the MAP optimization, including the Armijo Rule

Derivation of the Spatial Sampling Rate

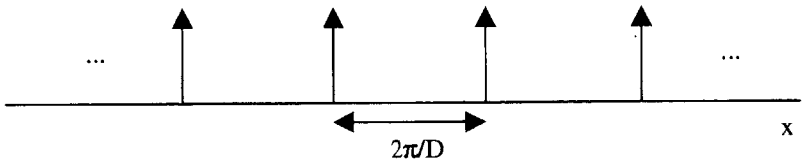
In Figure 3 it is stated that f_p, the first null of the pixel transfer function, is also the highest possible sampling rate. This is true of most imagers with 2-dimensional focal plane arrays. Exceptions can be found with scanning arrays, and these exceptions are discussed below.

To show that f_p is also the highest possible sampling rate, a one-dimensional argument is shown here. The extrapolation to two dimensions is straightforward.

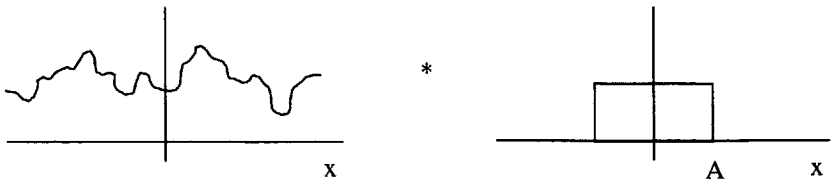
Sampling is often idealized as multiplication by a train of impulse functions (see [Oppenheim 1997] for a complete treatment of sampling). In the Fourier Transform domain, the multiplication of the signal (intensity distribution) and the impulse train becomes a convolution of their respective Fourier transforms. The Fourier transform of an impulse train is an impulse train of different amplitude and period. If the samples in the spatial domain differ by distance D,



then the distance between the impulses in the Fourier transform is $2\pi/D$, which is the spatial sampling rate:

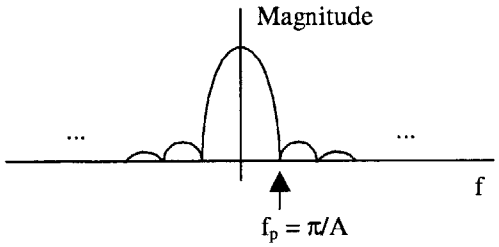


(All amplitudes are ignored in this derivation for simplicity since they do not affect the frequency relationships that are being demonstrated).
In a solid-state imager, the impulse train by itself is a poor model of the sampling process; intensity cannot be easily measured at infinitesimal points. Instead, a photodetecting element has finite size, and its output is related to the average (or total) intensity over the finite pixel area. A model that produces this averaging effect begins with a convolution of the signal with a pixel aperture.



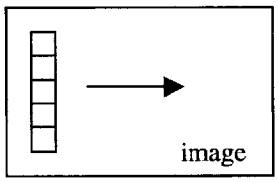
If one multiplies the result of the convolution by an impulse train, each sample gives the average value of the intensity over the pixel area that is centered on the sampling point.

To see that f_p is equal to the spatial sampling rate, examine the Fourier transform of the pixel aperture (the boxcar function above). Its Fourier transform is a sinc function with its first null at the frequency $f_p = \pi/A$:



The pixels of a 2-dimensional array cannot overlap, so it must be true that $2A <= D$ (the pixel spacing must at least as large as the pixel width). It is possible for $2A < D$, and this corresponds to fill factors of less than 100%. Even with $2A = D$ (a 100% fill factor), however, we see that the spatial sampling rate, $2\pi/D = 2\pi/2A = \pi/A = f_p$. Thus in the best case, the spatial sampling rate is f_p. If the fill factor is less than 100%, then the spatial sampling rate is less than f_p.

The statements above apply to 2-dimensional arrays of photodetectors. In Chapter 2 scanning arrays are also mentioned, in which the photodetectors are a 1-dimensional array and the second dimension is created by scanning the array across an image in time:



In this case, the pixels are sampled at regular time intervals to create a 2-dimensional image. Notice that the sampling rate can be set to frequencies higher than fp; this corresponds to sampling the pixels before they've shifted an entire pixel width from the previous sample. Thus scanning arrays do not face the same physical limitation on sampling rate in the scanning direction.

Derivation of the MAP optimization including the Armijo Rule

As stated in Chapter 3, the MAP derivation closely follows the RMMSE derivation. The MAP derivation is outlined here.

The MAP cost function given in Equation 15 is:

$$\hat{z} = \arg\min_{z} \frac{1}{2}(y - Wz)^T (y - Wz)$$

$$+ \frac{\lambda}{2} \sum_i \rho_\alpha((B_1 z)_i) + \rho_\alpha((B_2 z)_i) + \rho_\alpha((B_3 z)_i) + \rho_\alpha((B_4 z)_i)$$

The iterative optimization is:

$$z^{n+1} = z^n - \varepsilon^n g^n \quad .$$

The gradient, $\boldsymbol{g^n}$, is given by:

$$g^n = W^T(Wz - y) + \lambda\left(B_1^T f((B_1 z),\alpha) + B_2^T f((B_2 z),\alpha) + B_3^T f((B_3 z),\alpha) + B_4^T f((B_4 z),\alpha)\right) ,$$

where $f(\boldsymbol{v},\alpha)$ returns the vector $\boldsymbol{v}$, except every element of $\boldsymbol{v}$ that is greater than α is replaced with α, and every element less than $-\alpha$ is replaced by $-\alpha$.

The stepsize ε is then calculated by the Armijo Rule. The Armijo Rule guarantees convergence; its development is outlined in [Bertsekas 1995]. The rule operates according to the following steps:

* Choose an initial guess ε = s and choose a reduction factor β
* If C(z) – C(zk – $\varepsilon^k g^k$) ≥ a threshold value, then use this ε
* Otherwise set $\varepsilon = \varepsilon\beta$ and repeat the above step.

The threshold value is defined as $-\sigma\varepsilon\nabla C(z^k)'d^k$, where

$d^k = -\nabla C(z^k)$
σ = a parameter, we use 0.01
β = a paramter, we use 0.2

The Armijo stepsize is used to generate z^{k+1}, and the entire operation iterates until C(z) converges.

Appendix B – Matlab Code

This Appendix contains the Matlab code for all the algorithms and comparisons used in this thesis. The following functions are found below:

Comparison Functions:

ALIGN	- Used in the comparison process. Takes a truth image and a reconstruction and aligns them properly. The regions that overlap are passed to the COMPARE function (see below).
COMPARE	- Takes two images from ALIGN and calculates the mathematical metrics.

Super Resolution Algorithms:

MAP	- The maximum *a posteriori* implementation for uniform offsets.
MAPT	- The maximum *a posteriori* implementation for nonuniform offsets.
POCS	- The projection onto convex sets implementation for uniform offsets.
RMMSE	- The regularized minimum mean squared error implementation for uniform offsets.
RMMSET	- The RMMSE implementation for nonuniform offsets.
HARDIE	- Averages the offset data to produce a smooth initial estimate for the above algorithms.

Benchmark Components:

MINTERP	- Multiple frame interpolation algorithm.
SPECINTERP	- The multiple frame interpolation for the specific case of 2 frames with horizontal offsets
SPEC2INTERP	- The multiple frame interpolation for the special case of 2 frames with diagonal offsets
IP	- The inverse filtering algorithm that follows multi-frame interpolations (algorithm devleoped by Irani and Peleg [Irani 1991]).
CIF	- The inverse filtering algorithm used in the single frame case

Before the code is listed, some of the common data formats are summarized. Most of the algorithms require knowledge of the offsets between the low resolution frames. These offsets are stored in an array named 'R' or 'R_LR' (R is for Registration, and the optional LR is to clarify that the data is in units of Low Resolution Frames). The 'R' array has two columns and as many rows as there are images. Row one always corresponds to image one and has values [0 0]. Each subsequent row gives the offset of the current frame from the first image, and the data is stored as a fraction of the low-resolution pixel size for the row displacement and column displacement. Thus, if frame 2 is offset one half of a low-resolution pixel in the row direction and three quarters of an LR pixel in the column direction, row 2 of the 'R' array would be [0.5 0.75].

Another common data set used by the functions below is the Spatial Response Function. This data is stored in a two-dimensional array and is usually called 'psf,' despite the inconsistency with the terminology mentioned in Chapter 2. Unless stated otherwise, the 'psf' variable contains samples of the SRF at the sampling rate of the desired reconstruction.

Finally, the images themselves are stored in a 3-dimensional array (M, N, K), where M is the row, N is the column, and K is the frame.

ALIGN.M

Used in computing metrics. Align ensures that the two images are aligned correctly, and then calls compare.m, which produces the actual numeric metrics.

```
function out = align(orig, new);

% Shifts 2 images into alignment and compares using COMPARE.M
%
%                       OUT = ALIGN(ORIG, NEW)
%
% ORIG, NEW are images at the same sampling rates, and can be different sizes or misaligned.
% OUT is comparison data from COMPARE.M

% Create a cross-correlation
d = xcorr2(new,orig);

%% Find point with highest correlation.
%% The row is stored as 'r', the column as 'c'.
[temp, c] = max(max(d));
[temp, r] = max(max(d'));

%% Some size data for later
[rows_o, cols_o] = size(orig);
[rows_n, cols_n] = size(new);

% Calculate th upper left pixel of ORIG, relative to NEW
UL = [(r-rows_o+1), (c-cols_o+1)];

%% Set NEW and ORIG to just the aligned areas that overlap
new = new(max(UL(1),1):min(r,rows_n), max(UL(2),1):min(c,cols_n));
orig = orig(max(1,2-UL(1)):min(rows_n-UL(1)+1,rows_o), max(1,2-UL(2)):min(cols_n-UL(2)+1,cols_o));

%% Error out if there is a problem
if size(new) ~= size(orig)
  error('Algorithm failure: size mismatch');
end

%% Do the numerical comparisons by calling compare.m
out = compare(orig,new);

%% Also, do the same comparisons on the images, excluding the image boundaries
%% (5 pixels around the entire images) to eliminate possible boundary problems
[r,c] = size(orig);
out = [out; compare(orig(5:(r-5),5:(c-5)),new(5:(r-5),5:(c-5)))];

% End of function ALIGN.M
```

COMPARE.M

Used in computing metrics. Is called by ALIGN.M and returns a vector of metrics.

```
function out = compare(I1,I2)
% Compares two images using NMSE and NLMSE metrics
%
%    out = compare(I1,I2)
% Inputs I1, I2 are images
% OUT is a column vector of metrics:
%                       [       NMSE
%                               NLMSE (Laplacian)]

% Create a difference image
```

```
diff = I1 - I2;
[rows, cols] = size(diff);

% Energy of Image1 (used in normalizations below)
energyI1 = sum(sum(I1.^2));

% Mean Squared Error
MSE = sum(sum(diff.^2));                                    % Mean Square Error
NMSE = MSE / energyI1;                                      % NORMALIZED MSE  (from Pratt p.182)

% "Laplacian" MSE (from Pratt p.182)
kernel = [0 1 0; 1 -4 1; 0 1 0];
G1 = conv2(I1,kernel,'valid');
G2 = conv2(I2,kernel,'valid');
NLMSE = (sum(sum((G1 - G2).^2)))/(sum(sum(G1.^2)));

out = [NMSE; NLMSE];

% End of Function COMPARE.M
```

MAP.M

This implements the Maximum a posteriori restoration for all but the 4 and 8 frame random offsets.

```
function [out,COST] = map(y,R_LR,psf,lambda,nn,initial_image,weights)

% MAP algorithm with Huber-Markov Prior (Stevenson, Hardie)
%
%                    [OUT,COST] = map(Y,R_LR,PSF,LAMBDA,N,INIT_IMAGE,WEIGHTS)
%
% Y is the low res image sequence (MxNxP)
% R is the low res registration data (standard form)
% PSF is at the desired (high) resolution
% LAMBDA is the regularization parameter
% N is the number of iterations
% INIT_IMAGE is the first guess: 1=const,[2=hardie]
% WEIGHTS is a weights array calculated and saved by RMMSE.M
% OUT is the final high resolution image

% This version is designed to run as Matlab code; it vectorizes as many operations as possible.

global size_z size_y size_psf psf_center GAIN gridoffset_HR R_HR;

% Constants
GAIN = 2;  % Resolution increase in each dimension
alpha = 1;  % Huber-Markov Parameter; common value from Stevenson paper

% Set PSF_CENTER
psf_center = ceil(size(psf)/2);
size_psf = size(psf);

% Create a high resolution grid, convert all offsets to HR pixels
gridoffset_LR = [min(R_LR(:,1)), min(R_LR(:,2))];
R_HR = R_LR.*GAIN;
gridoffset_HR = gridoffset_LR.*GAIN;

[size_y(1) size_y(2) n_images] = size(y);

% Initial estimate of the high-res image
if nargin < 6
  initial_image = 2  % Default
end;
if initial_image == 1
  % Constant image
  size_z(1) = max(R_HR(:,1)-min(R_HR(:,1))+GAIN*size_y(1))-1;
  size_z(2) = max(R_HR(:,2)-min(R_HR(:,2))+GAIN*size_y(2))-1;
  average_value = sum(sum(y(:,:,1)))/size_y(1)/size_y(2);
  z = average_value*ones(size_z(1),size_z(2));
```

```
  clear average_value;
else
  % Default is same as initial_image = 'hardie' -- Nearest-neighbor interpolation
  z = hardie(y,R_LR,psf,1/300,'1995');
end; % of initial_image selection

% Begin gradient descent iteration here
number_of_iterations = nn;
for qq=1:number_of_iterations,

% Apply imaging process to HR estimate to get a LR estimate
sim = conv2(z,psf,'same');
for k=1:n_images,
  ri = (1-gridoffset_HR(1)+R_HR(k,1)):GAIN:(GAIN*size_y(1)-gridoffset_HR(1)+R_HR(k,1));
  ci = (1-gridoffset_HR(2)+R_HR(k,2)):GAIN:(GAIN*size_y(2)-gridoffset_HR(2)+R_HR(k,2));
  ysim(:,:,k) = sim(ri,ci);
end;
clear ri ci sim;  % Clear Memory space

% MSE between SIM and ACTUAL LR images
sum1 = y - ysim;

% Calculate the other common sums (4 different sums for Huber-Markov)
beta_1 = [1 -2 1];
beta_2 = [0.5 0 0; 0 -1 0; 0 0 0.5];
beta_3 = [1; -2; 1];
beta_4 = [0 0 0.5; 0 -1 0; 0.5 0 0];
sum2_1 = conv2(z,beta_1,'same');
sum2_2 = conv2(z,beta_2,'same');
sum2_3 = conv2(z,beta_3,'same');
sum2_4 = conv2(z,beta_4,'same');

% Calculate COST and display
COST = 0.5*sum(sum(sum(sum1.^2))) + lambda/2*(rho(sum2_1,alpha) + ...
  rho(sum2_2,alpha) + rho(sum2_3,alpha) + rho(sum2_4,alpha))

% Compute Gradients
g = weights*reshape(sum1,prod(size(sum1)),1);
g = reshape(g,size_z(1),size_z(2));                    % From vector in matrix form
% Special derivatives for piecewise cost function are computed
% by function GR(adient)
g = -g + lambda * (gr(sum2_1,beta_1,alpha) + gr(sum2_2,beta_2,alpha) ...
  + gr(sum2_3,beta_3,alpha) + gr(sum2_4,beta_4,alpha));

% Compute epsilon using the Armijo Rule
s = 10;
sigma = 1e-2;
beta = 1/5;
% Initialize
epsilon = s / beta;
K = sigma * sum(sum(g.^2));
new_cost = COST;
% Armijo Loop
while (COST - new_cost) < (epsilon*K)
  epsilon = epsilon * beta;
  % Calculation of NEW_COST   (This is fairly long, but compare it to the original COST computation above)
        temp = conv2((z-epsilon*g),psf,'same');
        for k=1:n_images,
        ri = (1-gridoffset_HR(1)+R_HR(k,1)):GAIN:(GAIN*size_y(1)-gridoffset_HR(1)+R_HR(k,1));
        ci = (1-gridoffset_HR(2)+R_HR(k,2)):GAIN:(GAIN*size_y(2)-gridoffset_HR(2)+R_HR(k,2));
        ysim(:,:,k) = temp(ri,ci);
        end;
        clear ri ci temp;
        % MSE between SIM and ACTUAL LR images
        sum1 = y - ysim;

        % Calculate the other common sums (4 different sums for Huber-Markov)
        sum2_1 = conv2((z-epsilon*g),beta_1,'same');
        sum2_2 = conv2((z-epsilon*g),beta_2,'same');
        sum2_3 = conv2((z-epsilon*g),beta_3,'same');
```

```
        sum2_4 = conv2((z-epsilon*g),beta_4,'same');

        % Calculate COST and display
        new_cost = 0.5*sum(sum(sum(sum1.^2))) + lambda/2*(rho(sum2_1,alpha) + ...
 rho(sum2_2,alpha) + rho(sum2_3,alpha) + rho(sum2_4,alpha));

end;
% End of Armijo Loop - Stepsize Epsilon has been chosen

% Update Z  (38)
z = z - epsilon*g;
if abs(COST - new_cost)/COST < 1e-5
  break;    % Break out of loop if we're really close to the minimum
end;

% Loop and iterate again
end;

out = z;

%%%%%%%%%% START FUNCTION RHO %%%%%%%%%%
function out = rho(x,alpha)

% See cost function in Stevenson
temp = abs(x) < alpha;
out = sum(sum((temp.*(x.^2)) + (not(temp).*(2*alpha*abs(x)-alpha^2))));

%%%%%%%%%% END FUNCTION RHO %%%%%%%%%%

%%%%%%%%%% START FUNCTION GR %%%%%%%%%%
function out = gr(x,beta,alpha)

% See my gradient calculations
out = x;
out(find(x > alpha)) = alpha;
out(find(x < alpha)) = -alpha;
out = conv2(out,beta,'same');

%%%%%%%%%% END FUNCTION GR %%%%%%%%%%

% End of function MAP.M
```

MAPT.M

This implements the Maximum a posteriori restoration for the 4 and 8 frame random offsets. The code is very similar to MAP.M, except for the handling of the different offsets.

```
function [out,COST] = map(y,R_LR,psf,lambda,nn,initial_image,weights)
% MAP algorithm with Huber-Markov Prior (Stevenson, Hardie)
%
%                   [OUT,COST] = map(Y,R_LR,PSF,LAMBDA,N,INIT_IMAGE,WEIGHTS)
%
% Y is the low res image sequence (MxNxP)
% R is the low res registration data (standard form)
% PSF is at the desired (high) resolution
% LAMBDA is the regularization parameter
% N is the number of iterations
% INIT_IMAGE is the first guess: 1=const,[2=hardie]
% WEIGHTS is a weights array that has been calculated and saved by RMMSET.M
% OUT is the final high resolution image

% This version is designed to run as Matlab code; it vectorizes as many operations as possible.

global size_z size_y size_psf psf_center GAIN gridoffset_HR R_HR;

% Resolution increase in each direction
GAIN = 2;
```

```
alpha = 1;  % Huber-Markov Parameter; common value from Stevenson paper

% Set PSF_CENTER
psf_center = ceil(size(psf)/2);
size_psf = size(psf);

% Create a high resolution grid, convert all offsets to HR pixels
gridoffset_LR = [min(R_LR(:,1)), min(R_LR(:,2))];
R_HR = R_LR.*GAIN;
gridoffset_HR = gridoffset_LR.*GAIN;

[size_y(1) size_y(2) n_images] = size(y);

% Initial estimate of the high-res image
if nargin < 6
  initial_image = 2   % Default
end;
if initial_image == 1
  % Constant image
  size_z(1) = max(R_HR(:,1)-min(R_HR(:,1))+GAIN*size_y(1))-1;
  size_z(2) = max(R_HR(:,2)-min(R_HR(:,2))+GAIN*size_y(2))-1;
  average_value = sum(sum(y(:,:,1)))/size_y(1)/size_y(2);
  z = average_value*ones(size_z(1),size_z(2));
  clear average_value;

else
  % Default is same as initial_image = 'hardie'
  % Nearest-neighbor interpolation
  % CHANGED FOR 8RANDOM
  %z = hardie(y,R_LR,psf,1/300,'1995');
  z = minterp(y,R_LR);
end; % of initial_image selection

% Begin gradient descent iteration here
number_of_iterations = nn;
for qq=1:number_of_iterations,

 % Apply imaging process to HR estimate to get a LR estimate
 ysim = weights'*reshape(z,prod(size(z)),1);
 ysim = reshape(ysim,size_y(1),size_y(2),n_images);
 % MSE between SIM and ACTUAL LR images
 sum1 = y - ysim;

 % Calculate the other common sums (4 different sums for Huber-Markov)
 beta_1 = [1 -2 1];
 beta_2 = [0.5 0 0; 0 -1 0; 0 0 0.5];
 beta_3 = [1; -2; 1];
 beta_4 = [0 0 0.5; 0 -1 0; 0.5 0 0];
 sum2_1 = conv2(z,beta_1,'same');
 sum2_2 = conv2(z,beta_2,'same');
 sum2_3 = conv2(z,beta_3,'same');
 sum2_4 = conv2(z,beta_4,'same');

 % Calculate COST and display
 COST = 0.5*sum(sum(sum(sum1.^2))) + lambda/2*(rho(sum2_1,alpha) + ...
  rho(sum2_2,alpha) + rho(sum2_3,alpha) + rho(sum2_4,alpha))

 % Compute Gradients
 g = weights*reshape(sum1,prod(size(sum1)),1);
 g = reshape(g,size_z(1),size_z(2));                    % From vector in matrix form
 % Special derivatives for piecewise cost function are computed
 % by function GR(adient)
 g = -g + lambda * (gr(sum2_1,beta_1,alpha) + gr(sum2_2,beta_2,alpha) ...
  + gr(sum2_3,beta_3,alpha) + gr(sum2_4,beta_4,alpha));

 % Compute epsilon using the Armijo Rule
 s = 10;
 sigma = 1e-2;
 beta = 1/5;
 % Initialize
```

```
epsilon = s / beta;
K = sigma * sum(sum(g.^2));
new_cost = COST;
% Armijo Loop
while (COST - new_cost) < (epsilon*K)
 epsilon = epsilon * beta;
 % Calculation of NEW_COST
 % (This is fairly long, but compare it to the original COST computation above)
        ysim = weights'*reshape((z-epsilon*g),prod(size(z)),1);
 ysim = reshape(ysim,size_y(1),size_y(2),n_images);

        % MSE between SIM and ACTUAL LR images
        sum1 = y - ysim;

        % Calculate the other common sums (4 different sums for Huber-Markov)
        sum2_1 = conv2((z-epsilon*g),beta_1,'same');
        sum2_2 = conv2((z-epsilon*g),beta_2,'same');
        sum2_3 = conv2((z-epsilon*g),beta_3,'same');
        sum2_4 = conv2((z-epsilon*g),beta_4,'same');

        % Calculate COST and display
        new_cost = 0.5*sum(sum(sum(sum1.^2))) + lambda/2*(rho(sum2_1,alpha) + ...
 rho(sum2_2,alpha) + rho(sum2_3,alpha) + rho(sum2_4,alpha));
end;
% End of Armijo Loop - Stepsize Epsilon has been chosen

% Update Z  (38)
z = z - epsilon*g;

if abs(COST - new_cost)/COST < 1e-5
  break;    % Break out of loop if we're really close to the minimum
end;

% Loop and iterate again
end;

out = z;

%%%%%%%%%% START FUNCTION RHO %%%%%%%%%%
function out = rho(x,alpha)

% See cost function in Stevenson
temp = abs(x) < alpha;
out = sum(sum((temp.*(x.^2)) + (not(temp).*(2*alpha*abs(x)-alpha^2))));

%%%%%%%%%% END FUNCTION RHO %%%%%%%%%%

%%%%%%%%%% START FUNCTION GR %%%%%%%%%%
function out = gr(x,beta,alpha)

% See my gradient calculations
out = x;
out(find(x > alpha)) = alpha;
out(find(x < alpha)) = -alpha;
out = conv2(out,beta,'same');

%%%%%%%%%% END FUNCTION GR %%%%%%%%%%

% End of function MAPT.M
```

POCS.M

This implements the Projection Onto Convex Sets restoration for all but the 4 and 8 frame random offsets.

```
function [out, projections_this_iter] = pocs(y,R_LR,psf,delta,nn,initial_image)
% POCS restoration algorithm (Stark and Oskoui, 1989)
%
%                   [OUT, Final_projections] = pocs(Y,R_LR,PSF,DELTA,N,INIT_IMAGE)
```

```
%
% Y is the low res image sequence (MxNxP)
% R is the low res registration data (standard form)
% PSF is at the desired (high) resolution
% DELTA is a parameter. Set to 0.01 for no noise, approx 3sigma for noise
% N is the number of iterations
% INIT_IMAGE is the first guess: 1=const,[2=hardie]
% OUT is the final high resolution image

% This is a POCS formulation that follows closely the development
% in Stark and Oskoui 1989. At a later time, additional convex constraints
% may be added, and will be noted here. The current implementation must
% be iterative, because the nature of POCS is iterative and each projection
% must operate on the output of the previous projection. This is no (obvious)
% way to parallelize or vectorize the computations. That having been said,
% the iterations were made as efficient as possible.

GAIN = 2;

% Define sigma, which is the flipped PSF
sigma = flipud(fliplr(psf));
sigma_center = ceil(size(sigma)/2);
sigma_size = size(sigma);
sigma_squared = sum(sum(sigma.^2));

% Define E for use in Energy Constraint below
% E containts the energy of each LR frame, multiplied by GAIN^2
E = sum(sum(y.^2)) * GAIN^2;

% Create a high resolution grid, convert all offsets to HR pixels
gridoffset_LR = [min(R_LR(:,1)), min(R_LR(:,2))];
R_HR = R_LR.*GAIN;
gridoffset_HR = gridoffset_LR.*GAIN;

% Initial estimate of the high-res image
[size_y(1) size_y(2) n_images] = size(y);
if initial_image == 1
  % Constant image
  size_z(1) = max(R_HR(:,1)-min(R_HR(:,1))+GAIN*size_y(1))-1;
  size_z(2) = max(R_HR(:,2)-min(R_HR(:,2))+GAIN*size_y(2))-1;
  average_value = sum(sum(y(:,:,1)))/size_y(1)/size_y(2);
  z = average_value*ones(size_z(1),size_z(2));
  clear average_value;
else
  % Default is same as initial_image = 'hardie'
  % Nearest-neighbor interpolation
  z = hardie(y,R_LR,psf,1/300,'1995');
end; % of initial_image selection
size_z = size(z);

% Create a frame of zeros around the HR data
total_size = size_z + 20;          % Frame is 10 pixels wide (20 total)
if sigma_center > 11                    % If the PSF extends beyond the frame, error
  error('This implementation requires a smaller PSF');
end;
temp = zeros(total_size);
temp(11:10+size_z(1),11:10+size_z(2)) = z;  % Place z into the center of the frame
z = temp;
clear temp;

% BEGINNING OF PROJECTION ITERATIONS
number_of_iterations = nn;
for qq=1:number_of_iterations,
  projections_this_iter = 0;

  % For each LR sample value, we must do a projection
  for k=1:n_images,
    % For this particular LR image (k), ri and ci give the HR grid point
    % that corresponds to each LR pixel.
    ri = (1-gridoffset_HR(1)+R_HR(k,1)):GAIN:(GAIN*size_y(1)-gridoffset_HR(1)+R_HR(k,1));
```

```
        ci = (1-gridoffset_HR(2)+R_HR(k,2)):GAIN:(GAIN*size_y(2)-gridoffset_HR(2)+R_HR(k,2));

  for i=1:length(ri),
    for j=1:length(ci),
      % First, we compute (z,sigma).
    r = ri(i); c = ci(j);
    out_row_l = 10 + r - sigma_center(1) + 1;
    out_row_u = 10 + r + sigma_center(1) - 1;
    out_col_l = 10 + c - sigma_center(2) + 1;
    out_col_u = 10 + c + sigma_center(2) - 1;


    % Now the upper and lower limits have been set. Using these limits,
    % we multiply to obtain (z,sigma)
    z_sigma = z(out_row_l:out_row_u,out_col_l:out_col_u) .* sigma;
    z_sigma = sum(sum(z_sigma));
    % If z_sigma = realdata, we're done. Otherwise we adjust z.
    difference = y(i,j,k) - z_sigma;
    if abs(difference) > delta,
      z(out_row_l:out_row_u,out_col_l:out_col_u) = ...
        z(out_row_l:out_row_u,out_col_l:out_col_u) + ...
        (difference/sigma_squared)*sigma;
      projections_this_iter = projections_this_iter + 1;
    end;

        end; % j
        end;   % i
end;     % k

%% At this point, we have carried out the projections for all the LR pixel
%% data. Next, we will perform the projections for the other criteria, such
%% as positivity, etc.

% Finite support condition
temp = zeros(total_size);
temp(11:10+size_z(1),11:10+size_z(2)) = z(11:10+size_z(1),11:10+size_z(2));
z = temp; clear temp;

% Amplitude Constraint (positivity)
z = (z >=0) .* z;                    % Changes all negative values to zero

% Energy Constraint
% Note: E is defined before the main iteration, since it does not change
% We impose a constraint for each LR image.  See additional notes for an
% explanation of how the energy bounds were determined.
for k = 1:n_images,
  % Compute the HR pixels that bound the LR image (ignoring the zero-padded frame)
  upper_left = [1-gridoffset_HR(1)+R_HR(k,1)  1-gridoffset_HR(2)+R_HR(k,2)];
  lower_right = upper_left + [GAIN*(size_y(1)-1)  GAIN*(size_y(2)-1)];
  % Account for the zero-pad frame around z, and move in by the radius of the PSF
  upper_left = upper_left + [9 9] + sigma_center;
  lower_right = lower_right + [11 11] - sigma_center;
  % Compute the energy of this "smaller" portion of the HR image
  Ez = sum(sum(z(upper_left(1):lower_right(1),upper_left(2):lower_right(2)).^2));
  % if Ez > E then adjust, otherwise do nothing.
  if Ez > E(k),
    z = z * sqrt(E(k)/Ez);
  end;
end;

% Display how many updates occured (if 0, we've reached the intersection
% of all the Convex Constraint Sets, and we can quit).
projections_this_iter
if projections_this_iter == 0
  projections_this_iter = -qq;  % A hack to output how many it took
  break;  % break out of loop; we're done
end;

end;  % of entire iteration procedure
```

```
out = z(11:size_z(1)+10,11:size_z(2)+10);

% End of function POCS.M
```

POCST.M

This implements the Projection Onto Convex Sets restoration for the 4 and 8 frame random offsets.

```
function [out, projections_this_iter] = pocs(y,R_LR,psf,delta,nn,initial_image)
% POCS restoration algorithm (Stark and Oskoui, 1989)
%
%                    [OUT, Final_projections] = pocs(Y,R_LR,PSF,DELTA,N,INIT_IMAGE)
%
% Y is the low res image sequence (MxNxP)
% R is the low res registration data (standard form)
% PSF is at the desired (high) resolution
% DELTA is a parameter.  Set to 0.01 for no noise, approx 3sigma for noise
% N is the number of iterations
% INIT_IMAGE is the first guess: 1=const,[2=hardie]
% OUT is the final high resolution image

% This is a POCS formulation that follows closely the development
% in Stark and Oskoui 1989.  At a later time, additional convex constraints
% may be added, and will be noted here.  The current implementation must
% be iterative, because the nature of POCS is iterative and each projection
% must operate on the output of the previous projection.  This is no (obvious)
% way to parallelize or vectorize the computations.  That having been said,
% the iterations were made as efficient as possible.

GAIN = 2;

% Define sigma, which is the flipped PSF
sigma = flipud(fliplr(psf));
sigma_center = ceil(size(sigma)/2);
sigma_size = size(sigma);
%sigma_squared = sum(sum(sigma.^2));
psfs = interp2(psf,'cubic');

% Define E for use in Energy Constraint below
% E containts the energy of each LR frame, multiplied by GAIN^2
E = sum(sum(y.^2)) * GAIN^2;

% Create a high resolution grid, convert all offsets to HR pixels
gridoffset_LR = [min(R_LR(:,1)), min(R_LR(:,2))];
R_HR = R_LR.*GAIN;
gridoffset_HR = gridoffset_LR.*GAIN;

% Initial estimate of the high-res image
[size_y(1) size_y(2) n_images] = size(y);
if initial_image == 1
  % Constant image
  %size_z(1) = max(R_HR(:,1)-min(R_HR(:,1))+GAIN*size_y(1))-1;
  %size_z(2) = max(R_HR(:,2)-min(R_HR(:,2))+GAIN*size_y(2))-1;
  size_z(1) = GAIN*size_y(1);
  size_z(2) = GAIN*size_y(2);
  average_value = sum(sum(y(:,:,1)))/size_y(1)/size_y(2);
  z = average_value*ones(size_z(1),size_z(2));
  clear average_value;
else
  % Default is same as initial_image = 'hardie'
  % Nearest-neighbor interpolation
  % CHANGED FOR 8RANDOM
  %z = hardie(y,R_LR,psf,1/300,'1995');
  z = minterp(y,R_LR);
end; % of initial_image selection
size_z = size(z);

% Create a frame of zeros around the HR data
```

```
total_size = size_z + 20;              % Frame is 10 pixels wide (20 total)
if sigma_center > 11                   % If the PSF extends beyond the frame, error
  error('This implementation requires a smaller PSF');
end;
temp = zeros(total_size);
temp(11:10+size_z(1),11:10+size_z(2)) = z;   % Place z into the center of the frame
z = temp;
clear temp;

% BEGINNING OF PROJECTION ITERATIONS
number_of_iterations = nn;
for qq=1:number_of_iterations,
  projections_this_iter = 0;

  % For each LR sample value, we must do a projection
  for k=1:n_images,
    % For this particular LR image (k), ri and ci give the HR grid point
    % that corresponds to each LR pixel.
    ri = (1-gridoffset_HR(1)+R_HR(k,1)):GAIN:(GAIN*size_y(1)-gridoffset_HR(1)+R_HR(k,1));
          ci = (1-gridoffset_HR(2)+R_HR(k,2)):GAIN:(GAIN*size_y(2)-gridoffset_HR(2)+R_HR(k,2));

    for i=1:length(ri),
      for j=1:length(ci),
        % First, we compute (z,sigma).
      r = ri(i); c = ci(j);
      %%% ADDITION
      if (mod(r,1)==0) &  (mod(c,1)==0)
        sigma = flipud(fliplr(psf));
        out_row_l = 10 + r - sigma_center(1) + 1;
            out_row_u = 10 + r + sigma_center(1) - 1;
            out_col_l = 10 + c - sigma_center(2) + 1;
            out_col_u = 10 + c + sigma_center(2) - 1;
      elseif (mod(r,1)==0) & (mod(c,1)==.5)
        sigma = flipud(fliplr(psfs(1:2:17,2:2:17)));
        out_row_l = 10 + r - sigma_center(1) + 1;
            out_row_u = 10 + r + sigma_center(1) - 1;
            out_col_l = 10 + ceil(c) - sigma_center(2) + 1;
        out_col_u = 10 + floor(c) + sigma_center(2) - 1;
      elseif (mod(r,1)==.5) & (mod(c,1)==0)
        sigma = flipud(fliplr(psfs(2:2:17,1:2:17)));
        out_row_l = 10 + ceil(r) - sigma_center(1) + 1;
        out_row_u = 10 + floor(r)+ sigma_center(1) - 1;
        out_col_l = 10 + c - sigma_center(2) + 1;
            out_col_u = 10 + c + sigma_center(2) - 1;
      elseif (mod(r,1)==.5) & (mod(c,1)==.5)
        sigma = flipud(fliplr(psfs(2:2:17,2:2:17)));
        out_row_l = 10 + ceil(r) - sigma_center(1) + 1;
        out_row_u = 10 + floor(r)+ sigma_center(1) - 1;
        out_col_l = 10 + ceil(c) - sigma_center(2) + 1;
        out_col_u = 10 + floor(c) + sigma_center(2) - 1;
      else
        error('Offsets are not multiples of 1/4 LR pixel');
      end;

      sigma_squared = sum(sum(sigma.^2));

      % Now the upper and lower limits have been set.  Using these limits,
      % we multiply to obtain (z,sigma)
      z_sigma = z(out_row_l:out_row_u,out_col_l:out_col_u) .* sigma;
      z_sigma = sum(sum(z_sigma));
      % If z_sigma = realdata, we're done.  Otherwise we adjust z.
      difference = y(i,j,k) - z_sigma;
      if abs(difference) > delta,
        z(out_row_l:out_row_u,out_col_l:out_col_u) = ...
          z(out_row_l:out_row_u,out_col_l:out_col_u) + ...
          (difference/sigma_squared)*sigma;
        projections_this_iter = projections_this_iter + 1;
      end;

            end; % j
```

```
        end;   % i
end;    % k

%% At this point, we have carried out the projections for all the LR pixel
%% data. Next, we will perform the projections for the other criteria, such
%% as positivity, etc.

% Finite support condition
temp = zeros(total_size);
temp(11:10+size_z(1),11:10+size_z(2)) = z(11:10+size_z(1),11:10+size_z(2));
z = temp; clear temp;

% Amplitude Constraint (positivity)
z = (z >=0) .* z;                    % Changes all negative values to zero

% Energy Constraint
% Note: E is defined before the main iteration, since it does not change
% We impose a constraint for each LR image.  See additional notes for an
% explanation of how the energy bounds were determined.
for k = 1:n_images,
  % Compute the HR pixels that bound the LR image (ignoring the zero-padded frame)
  upper_left = [1-gridoffset_HR(1)+R_HR(k,1)  1-gridoffset_HR(2)+R_HR(k,2)];
  lower_right = upper_left + [GAIN*(size_y(1)-1)  GAIN*(size_y(2)-1)];
  % Account for the zero-pad frame around z, and move in by the radius of the PSF
  upper_left = floor(upper_left + [9 9] + sigma_center);
  lower_right = ceil(lower_right + [11 11] - sigma_center);
  % Compute the energy of this "smaller" portion of the HR image
  Ez = sum(sum(z(upper_left(1):lower_right(1),upper_left(2):lower_right(2)).^2));
  % if Ez > E then adjust, otherwise do nothing.
  if Ez > E(k),
    z = z * sqrt(E(k)/Ez);
  end;
end;

% Display how many updates occured (if 0, we've reached the intersection
  % of all the Convex Constraint Sets, and we can quit).
  projections_this_iter
  if projections_this_iter == 0
    projections_this_iter = -qq;
    break;  % break out of loop; we're done
  end;
end;  % of entire iteration procedure

out = z(11:size_z(1)+10,11:size_z(2)+10);

% End of function POCST.M
```

RMMSE.M

This implements the Regularized Minimum Mean Squared Error
restoration for all but the 4 and 8 frame random offsets.

```
function [out,weights] = rmmse(y,R_LR,psf,lambda,number_of_iterations,initial_image,weights)
% Regularized MMSE algorithm (Hardie 1998)
%
%                    [OUT,weights] = rmmse4(Y,R_LR,PSF,LAMBDA,N,INIT_IMAGE,WEIGHTS)
%
% Y is the low res image sequence (MxNxP)
% R is the low res registration data (standard form)
% PSF is at the desired (high) resolution
% LAMBDA is the regularization parameter
% N is the number of iterations
% INIT_IMAGE is the first guess: 1=const,[2=hardie]
% WEIGHTS, if specified, is an already-calculated weights array
% OUT is the final high resolution image

% This version is designed to run as Matlab code;
% it vectorizes as many operations as possible.
```

```
global size_z size_y size_psf psf_center GAIN gridoffset_HR R_HR;

% Resolution increase in each direction
GAIN = 2;

% Set PSF_CENTER
psf_center = ceil(size(psf)/2);
size_psf = size(psf);

% Create a high resolution grid, convert all offsets to HR pixels
gridoffset_LR = [min(R_LR(:,1)), min(R_LR(:,2))];
R_HR = R_LR.*GAIN;
gridoffset_HR = gridoffset_LR.*GAIN;

[size_y(1) size_y(2) n_images] = size(y);
% Initial estimate of the high-res image
if nargin < 6
  initial_image = 2   % Default
end;
if initial_image == 1
  % Constant image
  size_z(1) = max(R_HR(:,1)-min(R_HR(:,1))+GAIN*size_y(1))-3; %-1;
  size_z(2) = max(R_HR(:,2)-min(R_HR(:,2))+GAIN*size_y(2))-3; %-1;
  average_value = sum(sum(y(:,:,1)))/size_y(1)/size_y(2);
  z = average_value*ones(size_z(1),size_z(2));
  clear average_value;

else
  % Default is same as initial_image = 'hardie'
  % Nearest-neighbor interpolation
  z = hardie(y,R_LR,psf,1/300,'1995');
end; % of initial_image selection
size_z = size(z);

% Compute the WEIGHTS matrix and store it if it hasn't
% been passed as an argument
if nargin < 7,

% Convert the scalar LR_PIXEL to a row,column,image (m,n,k)
% (k,m,n are row vectors.  Their length is the total number of LR pixels.
% Any given index into k,m,n gives a triplet (m,n,k) for a single LR pixel)
for kk = 1:n_images
  for nn = 1:size_y(2)
    for mm = 1:size_y(1)
      scalar_pixel = (kk-1)*size_y(1)*size_y(2) + (nn-1)*size_y(1) + mm;
      k(scalar_pixel) = kk;
      m(scalar_pixel) = mm;
      n(scalar_pixel) = nn;
    end;
  end;
end;
% Convert from a low res (m,n,k) to a high res Row & Column
z_row = 1 - gridoffset_HR(1) + R_HR(k,1)' + GAIN*(m-1);
z_col = 1 - gridoffset_HR(2) + R_HR(k,2)' + GAIN*(n-1);

% Create a sparse matrix of weights
% Below this point is a hack used due to memory constraints.
% The WEIGHTS matrix is calculated piecewise and stored, then
% is reassembled below from the files.

filenames = 'abcdefghij'; % 10 of them
for j = 0:9,
weights = [];
for i = 1:prod(size(z))/10,
  temp = weight(z_row,z_col,i+prod(size(z))/10*j,psf);  % Column Vector
%           temp = weight(z_row,z_col,i,psf);
            weights = [weights temp]; % Store the transpose (more efficient)
end;
save(filenames(j+1),'weights');
```

```
display('One tenth complete');
clear weights;
end;

% Now the portions of the weights matrix are stored in MAT files.
% We need to load, concatenate horizontally, then transpose to create
% the final WEIGHTS matrix in the proper orientation.
temp = [];
for j=1:10,
  clear weights, load(filenames(j));
  temp = [temp weights];
end;
weights = temp';
clear temp;

end; % of NARGIN < 7  (WEIGHTS unspecified)

% Begin gradient descent iteration here
for qq=1:number_of_iterations,

% Apply imaging process to HR estimate to get a LR estimate
sim = conv2(z,psf,'same');
for k=1:n_images,
  ri = (1-gridoffset_HR(1)+R_HR(k,1)):GAIN:(GAIN*size_y(1)-gridoffset_HR(1)+R_HR(k,1));
  ci = (1-gridoffset_HR(2)+R_HR(k,2)):GAIN:(GAIN*size_y(2)-gridoffset_HR(2)+R_HR(k,2));
  ysim(:,:,k) = sim(ri,ci);
end;
clear ri ci sim;
% MSE between SIM and ACTUAL LR images
sum1 = y - ysim;

% Calculate the other common sum (with alpha)
alpha = [0 -0.25 0; -0.25 1 -0.25; 0 -0.25 0];
sum2 = conv2(z,alpha,'same');

% Display COST for user
COST = 0.5*sum(sum(sum(sum1.^2))) + lambda/2*sum(sum(sum2.^2))
%disp('Press Ctrl-C to terminate, any other key to iterate again');
%pause

% Compute Gradients (iterations are instantaneous without the next line)
g = weights*reshape(sum1,prod(size(sum1)),1);
g = reshape(g,size_z(1),size_z(2));                    % From vector in matrix form
g = -g + (lambda * conv2(sum2,alpha,'same'));  % Add 'alpha' term

% Compute gamma
sim = conv2(g,psf,'same');
for k=1:n_images,
  ri = (1-gridoffset_HR(1)+R_HR(k,1)):GAIN:(GAIN*size_y(1)-gridoffset_HR(1)+R_HR(k,1));
  ci = (1-gridoffset_HR(2)+R_HR(k,2)):GAIN:(GAIN*size_y(2)-gridoffset_HR(2)+R_HR(k,2));
  gamma(:,:,k) = sim(ri,ci);
end;
clear ci ri sim;

% Compute g_bar
g_bar = conv2(g,alpha,'same');

% Compute epsilon
epsilon = sum(sum(sum(-gamma.*sum1))) + lambda*sum(sum(g_bar.*sum2));
epsilon = epsilon / (sum(sum(sum(gamma.^2))) + lambda*sum(sum(g_bar.^2)));

% Update Z  (38)
z = z - epsilon*g;

% Loop and iterate again
end;

out = z;

%%%%%%%%%% START FUNCTION WEIGHT %%%%%%%%%%
```

```
function out = weight(z_row,z_col,hr_pixel,psf)
global size_z size_y size_psf psf_center gridoffset_HR R_HR GAIN;

% Convert the scalar HR_PIXEL into a Row & Column
r_col = ceil(hr_pixel/size_z(1))*ones(size(z_row));
r_row = 1 + mod(hr_pixel-1,size_z(1))*ones(size(z_row));
% If hr_pixel is close enough to influence lr_pixel, find weight from PSF
r_row = psf_center(1) + z_row - r_row;  % (Reuse variables)
r_col = psf_center(2) + z_col - r_col;

criteria = (r_row > 0) & (r_col > 0) & (r_row <= size_psf(1)) & (r_col <= size_psf(2));
scalar_arg = r_row(find(criteria)) + size_psf(1)*(r_col(find(criteria))-1);

out = sparse(find(criteria),ones(size(scalar_arg)),psf(scalar_arg),length(r_row),1);

%%%%%%%%%% END FUNCTION WEIGHT %%%%%%%%%%

% End of function RMMSE.M
```

RMMSET.M

This implements the Regularized Minimum Mean Squared Error restoration for the 4 and 8 frame random offsets.

```
function [out,weights] = rmmset(y,R_LR,psf,lambda,number_of_iterations,initial_image,weights)
% Regularized MMSE algorithm (Hardie 1998)
%
%                    [OUT,weights] = rmmset(Y,R_LR,PSF,LAMBDA,N,INIT_IMAGE,WEIGHTS)
%
% Y is the low res image sequence (MxNxP)
% R is the low res registration data (standard form)
% PSF is at the desired (high) resolution
% LAMBDA is the regularization parameter
% N is the number of iterations
% INIT_IMAGE is the first guess: 1=const,[2=hardie]
% WEIGHTS, if specified, is an already-calculated weights array
% OUT is the final high resolution image

% This version is designed to run as Matlab code;
% it vectorizes as many operations as possible.

global size_z size_y size_psf psf_center GAIN gridoffset_HR R_HR;

% Resolution increase in each direction
GAIN = 2;

% Set PSF_CENTER
psf_center = ceil(size(psf)/2);
size_psf = size(psf);

% Create a high resolution grid, convert all offsets to HR pixels
gridoffset_LR = [min(R_LR(:,1)), min(R_LR(:,2))];
R_HR = R_LR.*GAIN;
gridoffset_HR = gridoffset_LR.*GAIN;

[size_y(1) size_y(2) n_images] = size(y);
% Initial estimate of the high-res image
if nargin < 6
  initial_image = 2   % Default
end;
if initial_image == 1
  % Constant image
  size_z(1) = max(R_HR(:,1)-min(R_HR(:,1))+GAIN*size_y(1))-3; %-1;
  size_z(2) = max(R_HR(:,2)-min(R_HR(:,2))+GAIN*size_y(2))-3; %-1;
  average_value = sum(sum(y(:,:,1)))/size_y(1)/size_y(2);
  z = average_value*ones(size_z(1),size_z(2));
  clear average_value;

else
```

```
  % Default is same as initial_image = 'hardie'
  % Nearest-neighbor interpolation
  % CHANGED FOR 8RANDOM
  %z = hardie(y,R_LR,psf,1/300,'1995');
  z = minterp(y,R_LR);
end; % of initial_image selection
size_z = size(z);

% Compute the WEIGHTS matrix and store it if it hasn't
% been passed as an argument
if nargin < 7,

% Convert the scalar LR_PIXEL to a row,column,image (m,n,k)
% (k,m,n are row vectors.  Their length is the total number of LR pixels.
% Any given index into k,m,n gives a triplet (m,n,k) for a single LR pixel)
for kk = 1:n_images
  for nn = 1:size_y(2)
    for mm = 1:size_y(1)
      scalar_pixel = (kk-1)*size_y(1)*size_y(2) + (nn-1)*size_y(1) + mm;
      k(scalar_pixel) = kk;
      m(scalar_pixel) = mm;
      n(scalar_pixel) = nn;
    end;
  end;
end;
% Convert from a low res (m,n,k) to a high res Row & Column
z_row = 1 - gridoffset_HR(1) + R_HR(k,1)' + GAIN*(m-1);
z_col = 1 - gridoffset_HR(2) + R_HR(k,2)' + GAIN*(n-1);

% Create a sparse matrix of weights
% Below this point is a hack used due to memory constraints.
% The WEIGHTS matrix is calculated piecewise and stored, then
% is reassembled below from the files.

filenames = 'abcdefghij'; % 10 of them
for j = 0:9,
weights = [];
for i = 1:prod(size(z))/10,
  temp = weight(z_row,z_col,i+prod(size(z))/10*j,psf);  % Column Vector
%           temp = weight(z_row,z_col,i,psf);
            weights = [weights temp]; % Store the transpose (more efficient)
end;
save(filenames(j+1),'weights');
display('One tenth complete');
clear weights;
end;

% Now the portions of the weights matrix are stored in MAT files.
% We need to load, concatenate horizontally, then transpose to create
% the final WEIGHTS matrix in the proper orientation.
temp = [];
for j=1:10,
  clear weights, load(filenames(j));
  temp = [temp weights];
end;
weights = temp';
clear temp;

end; % of NARGIN < 7  (WEIGHTS unspecified)

% Begin gradient descent iteration here
for qq=1:number_of_iterations,

% Apply imaging process to HR estimate to get a LR estimate
ysim = weights'*reshape(z,prod(size(z)),1);
ysim = reshape(ysim,size_y(1),size_y(2),n_images);
% MSE between SIM and ACTUAL LR images
sum1 = y - ysim;

% Calculate the other common sum (with alpha)
```

```
alpha = [0 -0.25 0; -0.25 1 -0.25; 0 -0.25 0];
sum2 = conv2(z,alpha,'same');

% Display COST for user
COST = 0.5*sum(sum(sum(sum1.^2))) + lambda/2*sum(sum(sum2.^2))
%disp('Press Ctrl-C to terminate, any other key to iterate again');
%pause

% Compute Gradients (iterations are instantaneous without the next line)
g = weights*reshape(sum1,prod(size(sum1)),1);
g = reshape(g,size_z(1),size_z(2));                    % From vector in matrix form
g = -g + (lambda * conv2(sum2,alpha,'same'));  % Add 'alpha' term

% Compute gamma
gamma = weights'*reshape(g,prod(size(g)),1);
gamma = reshape(gamma,size_y(1),size_y(2),n_images);
% Compute g_bar
g_bar = conv2(g,alpha,'same');

% Compute epsilon
epsilon = sum(sum(sum(-gamma.*sum1))) + lambda*sum(sum(g_bar.*sum2));
epsilon = epsilon / (sum(sum(sum(gamma.^2))) + lambda*sum(sum(g_bar.^2)));

% Update Z  (38)
z = z - epsilon*g;

% Loop and iterate again
end;

out = z;

%%%%%%%%% START FUNCTION WEIGHT %%%%%%%%%
function out = weight(z_row,z_col,hr_pixel,psf)
global size_z size_y size_psf psf_center gridoffset_HR R_HR GAIN;

% Convert the scalar HR_PIXEL into a Row & Column
r_col = ceil(hr_pixel/size_z(1))*ones(size(z_row));
r_row = 1 + mod(hr_pixel-1,size_z(1))*ones(size(z_row));
% If hr_pixel is close enough to influence lr_pixel, find weight from PSF
r_row = psf_center(1) + z_row - r_row;  % (Reuse variables)
r_col = psf_center(2) + z_col - r_col;

%out = zeros(size(r_row));
criteria = (r_row > 0) & (r_col > 0) & (r_row <= size_psf(1)) & (r_col <= size_psf(2));
scalar_arg = r_row(find(criteria)) + size_psf(1)*(r_col(find(criteria))-1);
%out(find(criteria)) = psf(scalar_arg);

out = sparse(find(criteria),ones(size(scalar_arg)),psf(scalar_arg),length(r_row),1);

%%%%%%%%% END FUNCTION WEIGHT %%%%%%%%%

% End of function RMMSET.M
```

MINTERP.M

This is a multi-frame interpolation method that creates a high resolution grid, and then fills each grid point with the weighted sum of the 3 nearest data points.

```
function out = minterp(gactual,R_LR,hpsf)
% A multi-frame interpolation method
%
%                                F = hardie(G,R,HPSF)
%
% G ~ MxNxK, K= number of low res images
% R = registration matrix for low res images (see notes)
% HPSF = psf at desired resolution
% F = a single high-res image
```

```
GAIN = 2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This is a multi-frame interpolation method that creates a high
%% resolution grid, and then fills each grid point with the weighted
%% sum of the 3 nearest data points. Once the high res grid has been
%% filled, it can be inverse-filtered with the PSF if desired.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Create a list of high resolution offsets
R_HR = R_LR*GAIN;

[g_rows g_cols num_images] = size(gactual);
n = num_images;

for c=0:2*g_cols-1,
  for r=0:2*g_rows-1,
    ro = mod(r, GAIN);
    co = mod(c, GAIN);

    % First, compute distances from current offset to all real data
    % points in the current LR pixel and all adjacent pixels.
    dist_center = R_HR - ones(num_images,1)*[ro co];
    dist_left  = R_HR - ones(num_images,1)*[ro co+GAIN];
    dist_right = R_HR - ones(num_images,1)*[ro co-GAIN];
    dist_up         = R_HR - ones(num_images,1)*[ro+GAIN co];
    dist_down  = R_HR - ones(num_images,1)*[ro-GAIN co];

    % The above matrices contain differences in the row and column
    % direction... we want to convert this to a single scalar distance.
    dist_center = sqrt(dist_center(:,1).^2 + dist_center(:,2).^2);
    dist_left  = sqrt(dist_left(:,1).^2  + dist_left(:,2).^2);
    dist_right = sqrt(dist_right(:,1).^2 + dist_right(:,2).^2);
    dist_up    = sqrt(dist_up(:,1).^2    + dist_up(:,2).^2);
    dist_down  = sqrt(dist_down(:,1).^2  + dist_down(:,2).^2);

    % Now we wish to find the 3 closest real data points
    % (the 3 points with the smallest distances). Note that if any
    % point has distance=0, we can quit and assign that value.
    distances = [dist_center; dist_left; dist_right; dist_up; dist_down];

    if distances > 0.001
      % (No point is distance 0)
      [d1 p1] = min(distances); distances(p1) = inf;
      [d2 p2] = min(distances); distances(p2) = inf;
      [d3 p3] = min(distances);

      % Calculate weights from the distances
      w1 = 1/d1; w2 = 1/d2; w3 = 1/d3;

      switch floor((p1-1)/num_images)
      case 0, row_adjust = 0; col_adjust = 0;
      case 1, row_adjust = 0; col_adjust = -1;
      case 2, row_adjust = 0; col_adjust = 1;
      case 3, row_adjust = -1; col_adjust = 0;
      case 4, row_adjust = 1; col_adjust = 0;
      otherwise, error('Error with point p1');
      end;

      out(r+1,c+1) = w1*eval('gactual(floor(r/GAIN)+1+row_adjust, floor(c/GAIN)+1+col_adjust, p1-floor((p1-1)/n)*n)', '0');

      switch floor((p2-1)/num_images)
      case 0, row_adjust = 0; col_adjust = 0;
      case 1, row_adjust = 0; col_adjust = -1;
      case 2, row_adjust = 0; col_adjust = 1;
      case 3, row_adjust = -1; col_adjust = 0;
      case 4, row_adjust = 1; col_adjust = 0;
      otherwise, error('Error with point p2');
      end;
```

```
      out(r+1,c+1) = out(r+1,c+1) + w2*eval('gactual(floor(r/GAIN)+1+row_adjust, floor(c/GAIN)+1+col_adjust, p2-floor((p2-
1)/n)*n)', '0');

      switch floor((p3-1)/num_images)
      case 0, row_adjust = 0; col_adjust = 0;
      case 1, row_adjust = 0; col_adjust = -1;
      case 2, row_adjust = 0; col_adjust = 1;
      case 3, row_adjust = -1; col_adjust = 0;
      case 4, row_adjust = 1; col_adjust = 0;
      otherwise, error('Error with point p3');
      end;

      out(r+1,c+1) = out(r+1,c+1) + w3*eval('gactual(floor(r/GAIN)+1+row_adjust, floor(c/GAIN)+1+col_adjust, p3-floor((p3-
1)/n)*n)', '0');
      out(r+1,c+1) = out(r+1,c+1)/(w1+w2+w3);

    else
      % One of the distances is zero
      [d p] = min(distances);
      if d < 0,
        error('Error: negative distance found!');
      end;
      if p > num_images,
        error('Error: a point from another frame has distance zero!');
      end;
      out(r+1,c+1) = gactual(floor(r/GAIN)+1, floor(c/GAIN)+1, p);
    end;
  end;
end;
% We're done filling the high res grid.

% End of function MINTERP.M
```

SPECINTERP.M

A special interpolation function designed solely for the case of 2 frames with purely horizontal offset.

```
function out = specinterp(y);
%
% OUT = specinterp(y);
%
% y is a 2-frame data set with horizontal offset = 0.5 LR

[rows cols n_images] = size(y);

for k=0:cols-1,
  out(:,1+2*k) = interp(y(:,1+k,1),2);
  out(:,2+2*k) = interp(y(:,1+k,2),2);
end;
% End of function SPECINTERP.M
```

SPECINTERP2.M

A special interpolation function designed solely for the case of 2 frames with diagonal offset.

```
function out = spec2interp(y);
%
% OUT = specinterp(y);
%
% y is a 2-frame data set with diagonal offset = [0.5 0.5]

[rows cols n_images] = size(y);

for c=1:cols
  for r=1:rows
```

```
    out(1+2*(r-1),1+2*(c-1)) = y(r,c,1);
    out(2+2*(r-1),2+2*(c-1)) = y(r,c,2);

    temp1 = eval('0.25*y(r,c+1,1)','0');
    temp2 = eval('0.25*y(r-1,c,2)','0');
    out(1+2*(r-1),2+2*(c-1)) = 0.25*y(r,c,1) + 0.25*y(r,c,2) + temp1 + temp2;

    temp3 = eval('0.25*y(r+1,c,1)','0');
    temp4 = eval('0.25*y(r,c-1,2)','0');
    out(2+2*(r-1),1+2*(c-1)) = 0.25*y(r,c,1) + 0.25*y(r,c,2) + temp3 + temp4;

  end;
end;
% End of function SPECINTERP2.M
```

IP.M

This is an inverse filtering algorithm used for some benchmarks, as devised by Irani and Peleg 1989.

```
function fnew = ip(gactual,hpsf,c,n)
% Irani and Peleg's method, ONE frame
%
%                                   f = ip(g,hpsf,c,n)
% Input g is a single low-res image,
%                        hpsf is arbitrary (usually hpsf), and its relative
%        resolution gives the resolution increase
%                        c is a constant (tweek experimentally).
%            n is the number of iterations to run
% Output f is a single high-res image

fold = gactual;                     % Initial guess

for i=1:n
  gdiff = gactual - conv2(fold,hpsf,'same');           % Apply model, calculate diff
  fnew = fold + 1/c*conv2(gdiff,hpsf,'same');                  % Backproject diffs
  fold = fnew;
          % Do again
end;

% End function IP.M
```

HARDIE.M

This computes the high resolution initial guess that is used by many of the super resolution algorithms. It creates this guess by filling every point in the high resolution grid with the value of the nearest data point.

```
function f = hardie(gactual,R_LR,hpsf,nsr,method)
% Hardie's method(s) for restoration
%
%                                   F = hardie(G,R,HPSF,NSR,METHOD)
%
% G ~ MxNxK, K= number of low res images
% R = registration matrix for low res images (see notes)
% HPSF = psf at desired resolution
% NSR = the Noise-to-Signal ratio of the images
% METHOD = '1995,' '1997'
% F = a single high-res image

GAIN = 2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Hardie has published 3 main articles, each successively more complex. His 1995 method, the simplest, consists of
%% registering the offsets, and just choosing the nearest LR pixel to fill each HR pixel. His 1997 method takes the
%% result of Method 1 and applies a certain Wiener filter as a deconvolution operator. His 1998 method is completely
%% unrelated, and is not covered in this file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if ndims(gactual) == 2                                                    % Create a variable
that contains
  n_images = 1;
          % the # of low-res images available
else n_images = size(gactual);
  n_images = n_images(3);
end

%%%%%%%%%%%%%%%%%%
%% METHOD 1 - 1995

% Round the registration values to the nearest HR points
R_LR = R_LR - fix(R_LR);                        % offset = offset MOD pixelsize
R_HR = round((R_LR*GAIN));                      % offset = quantize(offset)

[g_rows g_cols junk] = size(gactual);
hard_data = zeros(GAIN);

for offset_r=1:GAIN,
  for offset_c=1:GAIN,                          % for each offset within a single LR pixel

    % First, find what frame(s) are in this bin
    relevent = [];
    for row=1:n_images,
      if R_HR(row,:) == [(offset_r-1) (offset_c-1)]
        relevent = [relevent, row];
      end;
    end;

    % Second, if this bin has any frames, fill all such bins
    if length(relevent) > 0
      hard_data(offset_r,offset_c) = 1;         % Mark this bin as legit data
      if length(relevent) == 1                  % Expected Case = 1 frame per bin
        for i=0:(g_rows-1),
            for j=0:(g_cols-1)
            point = (1+[i j]-fix(R_LR(relevent,:)));
            if (point>=1) & (point <= [g_rows g_cols])
              f(offset_r+i*GAIN,offset_c+j*GAIN) = gactual(point(1),point(2),relevent);
            end;
          end;
        end;
      else  % Other Cases = Multiple frames per bin
        error('Multiple frames per bin is not supported yet');
      end;
    end;
  end;
end;

% At this point, the HR image F has certain pixels filled, and many pixels empty (0). For each empty pixel, we fill it
% according to Hardie 1995, which copies the value from the nearest neighbor.

% Now, we go through again and fill the still-empty bins

[drow dcol] = find(hard_data)     % Find points with hard data
for offset_r=1:GAIN,
  for offset_c=1:GAIN,

    if hard_data(offset_r,offset_c) == 0        % IF bin is empty
      list = sqrt((drow - offset_r).^2 + (dcol - offset_c).^2);
      % (list gives the distance from the current bin to all the
      % hard_data bins... so just find the smallest, and fill)
      [junk indices] = min(list);
      % Copy hard data into empty cells
      for i=0:(g_rows-1),
          for j=0:(g_cols-1)
        f(offset_r+i*GAIN,offset_c+j*GAIN) = f(drow(indices(1))+i*GAIN,dcol(indices(1))+j*GAIN);
        end;
      end;
    end;
```

```
  end;
end;

%% At this point, the complete high-res image resides in 'f'
%% This is the end of Hardie 1995. If the command line specified '1995' method, we exit here. Otherwise, we continue...

if method == '1995'
  return;
end;

%%%%%%%%%%%%%%%%%%
%% METHOD 2 - 1997

if method == '1997'

% FFT must be at least this many points to ensure true linear convolution
sizef = size(f);
ll = max(sizef + size(hpsf));

if ll > 1024
  error('Image too big; would require a 2048x2048 or larger FFT');
elseif ll > 512
  ll = 1024;
elseif ll > 256
  ll = 512;
elseif ll > 128
  ll = 256;
else
  ll = 128;
end;

% Convert to Frequency Domain
F = fft2(f,ll,ll);
Hpsf = fft2(hpsf,ll,ll);

% Create Wiener Filter
Hw = conj(Hpsf) ./ (abs(Hpsf).^2 + nsr);

% Apply Filter (multiplication in frequency domain)
F = F .* Hw;

% Inverse Filter and keep the valid part
f = abs(ifft2(F));
f = f(1:sizef(1),1:sizef(2));

%% Now 'f' holds the 1997 version of the image.
end;

% End of function HARDIE.M
```

CIF.M

This function performs a Constrained Inverse Filter. It is used solely in the 1-frame case. First this inverse filter is run on the data, and then the result is cubic interpolated up to the high resolution grid (the interpolation is not in this file).

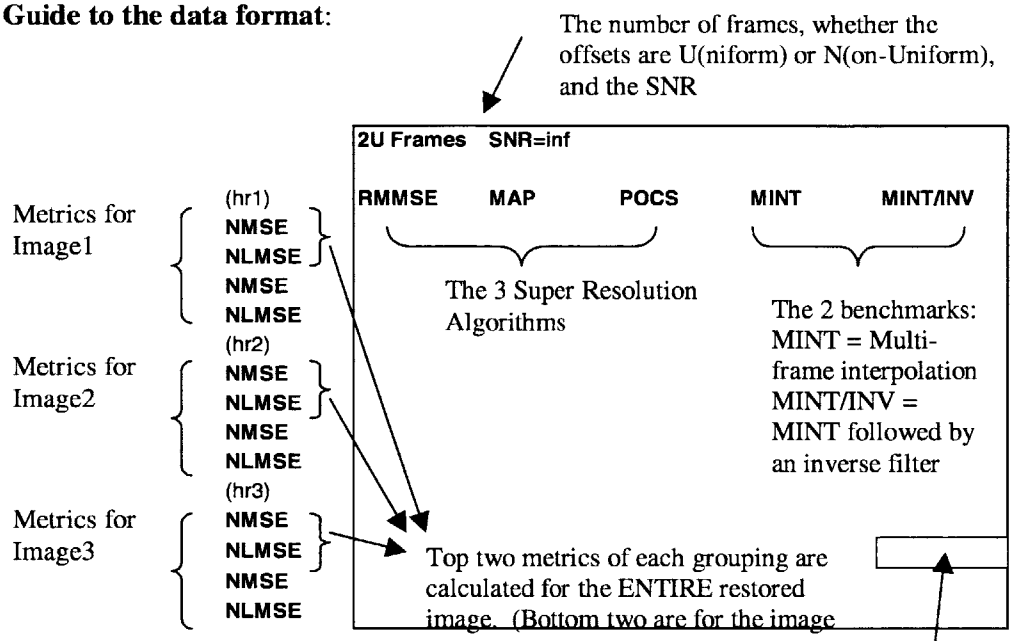
```
function out = cif(i, psf, low_limit)
% Perform a Constrained Inverse Filter on image
%
%                       OUT = cif(IMAGE, FULL_PSF, LOW_LIMIT)
%

if nargin == 2
  low_limit = 0.1;
end
hlow = psf;

% Choose FFT size
[rows cols] = size(i);
```

```
if ((rows < 246) & (cols < 246))
  fft_size = 256;
elseif ((rows < 502) & (cols < 502))
  fft_size = 512;
elseif ((rows < 1014) & (cols < 1014))
  fft_size = 1024;
else
  error('Input image too large.  Use partial image.');
end

% FFT, constrain, multiply, IFFT
Hlow = fft2(hlow,fft_size,fft_size);
I    = fft2(i,fft_size,fft_size);
Hcon = max(Hlow,(low_limit*Hlow./abs(Hlow)));
% possibly do something to the above line to smooth it...
clear Hlow;
Inew = I./Hcon;
clear I;                                % No longer need it, but we need to manage memory!
out = abs(ifft2(Inew));
out = out(1:rows,1:cols);


% End of function CIF.M
```

Appendix C – Complete Data

This Appendix contains the complete results.
The data is divided first into sections for each number of frames and SNR. For each section, the results of the metrics are shown for all algorithms and each of the three test images (hr1 = Rhode Island, hr2 = South Dakota, hr3 = Lena). Each metric is listed twice for every situation: the top number is the result of the metric applied to the entire image, and the bottom number is the result of the metric applied to the image without a border region 5 pixels wide.

Guide to the data format:



For example, the data in this square would correspond to:
Image = #3 = Lena
Metric = NLMSE on entire image
Algorithm = Multi-frame interpolation followed by inverse filter
Number of Frames = 2
Offsets between frames = uniform
SNR = infinity (no noisc)

	1 Frame SNR=inf					1 Frame SNR=300				
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV	RMMSE	MAP	POCS	MINT	MINT/IN
NMSE	4.29E-02	4.15E-02	4.19E-02	4.81E-02	5.25E-02	4.32E-02	4.16E-02	4.19E-02	4.81E-02	5.26E-02
NLMSE	9.49E-01	9.43E-01	9.40E-01	9.69E-01	9.67E-01	9.49E-01	9.44E-01	9.40E-01	9.69E-01	9.67E-01
NMSE	3.99E-02	4.04E-02	4.00E-02	4.43E-02	4.29E-02	3.99E-02	4.05E-02	4.02E-02	4.43E-02	4.29E-02
NLMSE	9.48E-01	9.48E-01	9.43E-01	9.69E-01	9.68E-01	9.48E-01	9.49E-01	9.43E-01	9.69E-01	9.68E-01
(hr2)										
NMSE	3.91E-02	3.56E-02	4.02E-02	4.77E-02	6.01E-02	3.95E-02	3.57E-02	4.02E-02	4.77E-02	6.02E-02
NLMSE	8.91E-01	8.60E-01	9.04E-01	9.21E-01	9.35E-01	8.92E-01	8.59E-01	9.05E-01	9.22E-01	9.35E-01
NMSE	3.64E-02	3.56E-02	3.79E-02	4.54E-02	5.13E-02	3.66E-02	3.56E-02	3.81E-02	4.54E-02	5.14E-02
NLMSE	8.66E-01	8.42E-01	8.92E-01	9.09E-01	9.22E-01	8.67E-01	8.40E-01	8.93E-01	9.09E-01	9.22E-01
(hr3)										
NMSE	9.11E-03	8.35E-03	1.07E-02	1.40E-02	2.11E-02	9.31E-03	7.85E-03	1.08E-02	1.41E-02	2.11E-02
NLMSE	8.55E-01	9.08E-01	9.93E-01	8.93E-01	9.18E-01	8.57E-01	8.81E-01	9.95E-01	8.94E-01	9.18E-01
NMSE	7.28E-03	8.01E-03	9.13E-03	1.03E-02	1.10E-02	7.32E-03	7.69E-03	9.19E-03	1.03E-02	1.10E-02
NLMSE	8.29E-01	8.89E-01	9.82E-01	8.86E-01	8.99E-01	8.30E-01	8.73E-01	9.85E-01	8.87E-01	8.99E-01

	1 frame SNR=150					1 frame SNR=75				
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV	RMMSE	MAP	POCS	MINT	MINT/IN
NMSE	4.34E-02	4.17E-02	4.23E-02	4.82E-02	5.26E-02	4.41E-02	4.23E-02	4.40E-02	4.87E-02	5.30E-02
NLMSE	9.50E-01	9.43E-01	9.44E-01	9.69E-01	9.68E-01	9.52E-01	9.45E-01	9.49E-01	9.70E-01	9.68E-01
NMSE	4.01E-02	4.06E-02	4.08E-02	4.44E-02	4.29E-02	4.08E-02	4.10E-02	4.28E-02	4.48E-02	4.33E-02
NLMSE	9.49E-01	9.49E-01	9.48E-01	9.69E-01	9.68E-01	9.51E-01	9.52E-01	9.53E-01	9.70E-01	9.69E-01
(hr2)										
NMSE	3.98E-02	3.59E-02	4.06E-02	4.79E-02	6.04E-02	4.12E-02	3.75E-02	4.22E-02	4.85E-02	6.11E-02
NLMSE	8.93E-01	8.59E-01	9.08E-01	9.22E-01	9.36E-01	9.02E-01	8.76E-01	9.25E-01	9.25E-01	9.38E-01
NMSE	3.69E-02	3.58E-02	3.86E-02	4.56E-02	5.15E-02	3.86E-02	3.73E-02	4.08E-02	4.63E-02	5.24E-02
NLMSE	8.69E-01	8.41E-01	8.96E-01	9.11E-01	9.22E-01	8.77E-01	8.58E-01	9.16E-01	9.14E-01	9.25E-01
(hr3)										
NMSE	9.46E-03	7.97E-03	1.09E-02	1.41E-02	2.12E-02	9.95E-03	8.45E-03	1.16E-02	1.43E-02	2.14E-02
NLMSE	8.60E-01	8.81E-01	9.99E-01	8.95E-01	9.19E-01	8.69E-01	8.93E-01	1.026	8.98E-01	9.21E-01
NMSE	7.46E-03	7.79E-03	9.33E-03	1.04E-02	1.11E-02	7.94E-03	8.25E-03	9.96E-03	1.06E-02	1.13E-02
NLMSE	8.32E-01	8.73E-01	9.88E-01	8.88E-01	9.00E-01	8.41E-01	8.85E-01	1.014	8.91E-01	9.01E-01

	2N Frames SNR=inf					2N Frames SNR=300				
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV	RMMSE	MAP	POCS	MINT	MINT/IN
NMSE	3.63E-02	3.70E-02	2.72E-02	4.48E-02	3.91E-02	3.63E-02	3.70E-02	3.27E-02	4.48E-02	3.92E-02
NLMSE	8.85E-01	8.92E-01	7.29E-01	9.31E-01	9.21E-01	8.85E-01	8.93E-01	8.40E-01	9.31E-01	9.21E-01
NMSE	3.44E-02	3.63E-02	2.74E-02	4.05E-02	3.76E-02	3.45E-02	3.64E-02	3.29E-02	4.05E-02	3.77E-02
NLMSE	8.88E-01	9.03E-01	7.42E-01	9.34E-01	9.23E-01	8.88E-01	9.04E-01	8.53E-01	9.34E-01	9.23E-01
(hr2)										
NMSE	3.13E-02	3.06E-02	2.39E-02	4.68E-02	3.65E-02	3.14E-02	3.06E-02	2.85E-02	4.69E-02	3.67E-02
NLMSE	8.04E-01	7.86E-01	6.89E-01	8.76E-01	8.54E-01	8.06E-01	7.86E-01	8.10E-01	8.78E-01	8.56E-01
NMSE	3.03E-02	3.07E-02	2.65E-02	4.09E-02	3.49E-02	3.05E-02	3.07E-02	3.11E-02	4.10E-02	3.50E-02
NLMSE	7.87E-01	7.72E-01	6.94E-01	8.63E-01	8.34E-01	7.89E-01	7.73E-01	8.08E-01	8.64E-01	8.36E-01
(hr3)										
NMSE	6.44E-03	5.02E-03	5.25E-03	1.29E-02	7.05E-03	6.50E-03	5.10E-03	6.50E-03	1.30E-02	7.10E-03
NLMSE	7.28E-01	6.98E-01	7.17E-01	8.11E-01	7.76E-01	7.30E-01	7.00E-01	8.51E-01	8.12E-01	7.78E-01
NMSE	5.10E-03	4.83E-03	5.34E-03	8.25E-03	5.94E-03	5.10E-03	4.90E-03	6.60E-03	8.30E-03	6.00E-03
NLMSE	6.99E-01	6.85E-01	7.01E-01	7.94E-01	7.44E-01	7.00E-01	6.87E-01	8.31E-01	7.95E-01	7.46E-01

	2N Frames	SNR=150				2N Frames	SNR=75			
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV	RMMSE	MAP	POCS	MINT	MINT/IN
NMSE	3.66E-02	3.72E-02	3.50E-02	4.50E-02	3.93E-02	3.75E-02	3.76E-02	3.82E-02	4.56E-02	4.04E-02
NLMSE	8.87E-01	8.94E-01	8.74E-01	9.32E-01	9.22E-01	8.94E-01	8.96E-01	9.04E-01	9.38E-01	9.32E-01
NMSE	3.47E-02	3.64E-02	3.53E-02	4.06E-02	3.78E-02	3.56E-02	3.68E-02	3.82E-02	4.12E-02	3.89E-02
NLMSE	8.90E-01	9.05E-01	8.87E-01	9.35E-01	9.24E-01	8.98E-01	9.09E-01	9.17E-01	9.43E-01	9.36E-01
(hr2)										
NMSE	3.18E-02	3.08E-02	3.04E-02	4.71E-02	3.72E-02	3.35E-02	3.20E-02	3.42E-02	4.79E-02	3.91E-02
NLMSE	8.10E-01	7.88E-01	8.38E-01	8.81E-01	8.63E-01	8.31E-01	8.04E-01	8.73E-01	8.93E-01	8.87E-01
NMSE	3.09E-02	3.09E-02	3.32E-02	4.12E-02	3.57E-02	3.27E-02	3.20E-02	3.70E-02	4.22E-02	3.77E-02
NLMSE	7.94E-01	7.77E-01	8.38E-01	8.69E-01	8.44E-01	8.14E-01	7.91E-01	8.70E-01	8.83E-01	8.70E-01
(hr3)										
NMSE	6.60E-03	5.20E-03	7.30E-03	1.30E-02	7.30E-03	7.20E-03	5.90E-03	8.50E-03	1.34E-02	8.00E-03
NLMSE	7.38E-01	7.07E-01	8.96E-01	8.17E-01	7.86E-01	7.60E-01	7.38E-01	9.45E-01	8.35E-01	8.20E-01
NMSE	5.30E-03	5.00E-03	7.30E-03	8.30E-03	6.10E-03	5.90E-03	5.60E-03	8.40E-03	8.70E-03	6.80E-03
NLMSE	7.09E-01	6.95E-01	8.76E-01	8.00E-01	7.54E-01	7.30E-01	7.23E-01	9.24E-01	8.18E-01	7.88E-01

	2U Frames	SNR=inf				2U Frames	SNR=300			
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV	RMMSE	MAP	POCS	MINT	MINT/IN
NMSE	2.87E-02	3.05E-02	2.48E-02	4.57E-02	3.98E-02	2.88E-02	3.07E-02	2.68E-02	4.57E-02	3.98E-02
NLMSE	8.31E-01	8.35E-01	7.70E-01	9.47E-01	9.56E-01	8.32E-01	8.39E-01	7.97E-01	9.47E-01	9.56E-01
NMSE	2.78E-02	2.96E-02	2.44E-02	4.02E-02	3.64E-02	2.79E-02	2.98E-02	2.63E-02	4.02E-02	3.65E-02
NLMSE	8.32E-01	8.32E-01	7.76E-01	9.36E-01	9.40E-01	8.33E-01	8.35E-01	8.02E-01	9.36E-01	9.40E-01
(hr2)										
NMSE	1.73E-02	1.51E-02	1.34E-02	4.62E-02	3.37E-02	1.75E-02	1.51E-02	1.59E-02	4.62E-02	3.39E-02
NLMSE	6.46E-01	5.90E-01	5.71E-01	9.18E-01	9.29E-01	6.49E-01	5.87E-01	6.20E-01	9.18E-01	9.31E-01
NMSE	1.69E-02	1.53E-02	1.38E-02	4.07E-02	3.14E-02	1.71E-02	1.53E-02	1.61E-02	4.08E-02	3.16E-02
NLMSE	6.18E-01	5.67E-01	5.54E-01	9.00E-01	8.88E-01	6.20E-01	5.65E-01	5.99E-01	9.00E-01	8.89E-01
(hr3)										
NMSE	4.61E-03	4.21E-03	4.00E-03	1.41E-02	9.06E-03	4.70E-03	4.30E-03	5.20E-03	1.41E-02	9.10E-03
NLMSE	6.96E-01	6.91E-01	6.83E-01	9.04E-01	9.28E-01	7.00E-01	6.97E-01	7.83E-01	9.06E-01	9.32E-01
NMSE	4.30E-03	4.20E-03	4.04E-03	9.24E-03	6.37E-03	4.40E-03	4.30E-03	5.10E-03	9.30E-03	6.40E-03
NLMSE	6.72E-01	6.84E-01	6.75E-01	8.57E-01	8.17E-01	6.76E-01	6.91E-01	7.63E-01	8.58E-01	8.20E-01

	2U Frames	SNR=150				2U Frames	SNR=75			
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV	RMMSE	MAP	POCS	MINT	MINT/IN
NMSE	2.91E-02	3.05E-02	2.88E-02	4.58E-02	4.00E-02	3.03E-02	3.12E-02	3.22E-02	4.61E-02	4.08E-02
NLMSE	8.35E-01	8.32E-01	8.20E-01	9.48E-01	9.57E-01	8.42E-01	8.38E-01	8.50E-01	9.51E-01	9.65E-01
NMSE	2.83E-02	2.97E-02	2.82E-02	4.03E-02	3.66E-02	2.94E-02	3.03E-02	3.15E-02	4.06E-02	3.75E-02
NLMSE	8.36E-01	8.32E-01	8.25E-01	9.37E-01	9.41E-01	8.45E-01	8.37E-01	8.56E-01	9.41E-01	9.50E-01
(hr2)										
NMSE	1.79E-02	1.55E-02	1.84E-02	4.63E-02	3.41E-02	2.01E-02	1.67E-02	2.28E-02	4.67E-02	3.53E-02
NLMSE	6.55E-01	5.94E-01	6.60E-01	9.20E-01	9.34E-01	6.83E-01	6.07E-01	7.10E-01	9.25E-01	9.51E-01
NMSE	1.76E-02	1.57E-02	1.87E-02	4.09E-02	3.19E-02	2.00E-02	1.69E-02	2.34E-02	4.14E-02	3.32E-02
NLMSE	6.27E-01	5.73E-01	6.38E-01	9.01E-01	8.92E-01	6.55E-01	5.85E-01	6.88E-01	9.07E-01	9.09E-01
(hr3)										
NMSE	4.90E-03	4.40E-03	6.30E-03	1.41E-02	9.30E-03	5.70E-03	5.30E-03	7.70E-03	1.44E-02	9.80E-03
NLMSE	7.10E-01	7.00E-01	8.42E-01	9.08E-01	9.38E-01	7.53E-01	7.52E-01	8.95E-01	9.23E-01	9.70E-01
NMSE	4.60E-03	4.40E-03	6.00E-03	9.30E-03	6.60E-03	5.40E-03	5.20E-03	7.30E-03	9.50E-03	7.20E-03
NLMSE	6.86E-01	6.94E-01	8.19E-01	8.61E-01	8.27E-01	7.27E-01	7.40E-01	8.72E-01	8.74E-01	8.58E-01

	4N Frames	SNR=inf				4N Frames	SNR=300			
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV	RMMSE	MAP	POCS	MINT	MINT/IN
NMSE	2.56E-02	2.62E-02	No Data	4.70E-02	4.14E-02	2.57E-02	2.62E-02	No Data	4.70E-02	4.15E-02
NLMSE	7.76E-01	7.54E-01		9.56E-01	9.71E-01	7.76E-01	7.54E-01		9.56E-01	9.72E-01
NMSE	2.46E-02	2.57E-02		4.12E-02	3.80E-02	2.47E-02	2.57E-02		4.12E-02	3.81E-02
NLMSE	7.75E-01	7.64E-01		9.47E-01	9.59E-01	7.76E-01	7.63E-01		9.47E-01	9.59E-01
(hr2)										
NMSE	1.54E-02	1.32E-02		4.87E-02	3.71E-02	1.56E-02	1.32E-02		4.87E-02	3.72E-02
NLMSE	5.90E-01	5.12E-01		9.38E-01	9.69E-01	5.94E-01	5.08E-01		9.39E-01	9.71E-01
NMSE	1.46E-02	1.20E-02		4.33E-02	3.55E-02	1.48E-02	1.20E-02		4.33E-02	3.56E-02
NLMSE	5.62E-01	4.81E-01		9.23E-01	9.39E-01	5.66E-01	4.78E-01		9.23E-01	9.41E-01
(hr3)										
NMSE	4.08E-03	4.76E-03		1.51E-02	9.44E-03	4.16E-03	4.83E-03		1.51E-02	9.47E-03
NLMSE	6.08E-01	6.03E-01		8.99E-01	9.18E-01	6.14E-01	6.10E-01		9.01E-01	9.20E-01
NMSE	3.39E-03	2.87E-03		9.71E-03	6.75E-03	3.47E-03	2.94E-03		9.72E-03	6.78E-03
NLMSE	5.81E-01	5.29E-01		8.57E-01	8.24E-01	5.88E-01	5.35E-01		8.58E-01	8.26E-01

	4N Frames	SNR=150				4N Frames	SNR=75			
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV	RMMSE	MAP	POCS	MINT	MINT/IN
NMSE	2.61E-02	2.65E-02	No Data	4.71E-02	4.17E-02	2.72E-02	2.71E-02	No Data	4.73E-02	4.22E-02
NLMSE	7.80E-01	7.58E-01		9.58E-01	9.74E-01	7.93E-01	7.66E-01		9.59E-01	9.79E-01
NMSE	2.50E-02	2.59E-02		4.13E-02	3.83E-02	2.62E-02	2.65E-02		4.14E-02	3.89E-02
NLMSE	7.80E-01	7.66E-01		9.49E-01	9.62E-01	7.95E-01	7.75E-01		9.51E-01	9.68E-01
(hr2)										
NMSE	1.60E-02	1.35E-02		4.88E-02	3.73E-02	1.78E-02	1.53E-02		4.94E-02	3.86E-02
NLMSE	6.01E-01	5.17E-01		9.40E-01	9.72E-01	6.34E-01	5.57E-01		9.49E-01	9.94E-01
NMSE	1.53E-02	1.24E-02		4.34E-02	3.57E-02	1.72E-02	1.41E-02		4.41E-02	3.71E-02
NLMSE	5.74E-01	4.89E-01		9.23E-01	9.41E-01	6.06E-01	5.29E-01		9.33E-01	9.63E-01
(hr3)										
NMSE	4.33E-03	5.04E-03		1.52E-02	9.63E-03	5.08E-03	5.98E-03		1.54E-02	1.01E-02
NLMSE	6.23E-01	6.23E-01		9.05E-01	9.31E-01	6.76E-01	7.07E-01		9.20E-01	9.57E-01
NMSE	3.62E-03	3.09E-03		9.82E-03	6.95E-03	4.39E-03	3.99E-03		9.96E-03	7.37E-03
NLMSE	5.95E-01	5.44E-01		8.62E-01	8.37E-01	6.49E-01	6.28E-01		8.78E-01	8.65E-01

	4U Frames	SNR=inf				4U Frames	SNR=300			
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV	RMMSE	MAP	POCS	MINT	MINT/IN
NMSE	2.24E-02	2.30E-02	2.60E-03	4.02E-02	3.83E-02	2.24E-02	2.30E-02	1.53E-02	4.02E-02	3.84E-02
NLMSE	7.21E-01	6.98E-01	1.22E-01	8.86E-01	9.27E-01	7.23E-01	6.95E-01	5.46E-01	8.85E-01	9.26E-01
NMSE	2.20E-02	2.31E-02	2.60E-03	3.64E-02	3.60E-02	2.21E-02	2.30E-02	1.53E-02	3.64E-02	3.61E-02
NLMSE	7.22E-01	7.09E-01	1.21E-01	8.86E-01	9.22E-01	7.23E-01	7.07E-01	5.54E-01	8.85E-01	9.22E-01
(hr2)										
NMSE	1.29E-02	9.60E-03	1.00E-03	3.94E-02	3.52E-02	1.31E-02	9.70E-03	9.90E-03	3.94E-02	3.54E-02
NLMSE	5.35E-01	4.24E-01	6.71E-02	8.08E-01	9.06E-01	5.39E-01	4.25E-01	4.42E-01	8.12E-01	9.11E-01
NMSE	1.26E-02	9.60E-03	1.10E-03	3.54E-02	3.39E-02	1.28E-02	9.70E-03	1.01E-02	3.55E-02	3.42E-02
NLMSE	5.08E-01	4.02E-01	6.68E-02	7.90E-01	8.61E-01	5.12E-01	4.03E-01	4.24E-01	7.94E-01	8.67E-01
(hr3)										
NMSE	3.10E-03	2.30E-03	2.00E-04	1.15E-02	8.30E-03	3.10E-03	2.40E-03	2.80E-03	1.16E-02	8.40E-03
NLMSE	5.44E-01	4.64E-01	6.91E-02	7.67E-01	8.38E-01	5.49E-01	4.73E-01	4.84E-01	7.71E-01	8.43E-01
NMSE	2.90E-03	2.30E-03	3.00E-04	7.50E-03	5.40E-03	3.00E-03	2.40E-03	2.60E-03	7.50E-03	5.50E-03
NLMSE	5.22E-01	4.52E-01	6.68E-02	7.34E-01	7.04E-01	5.27E-01	4.59E-01	4.62E-01	7.38E-01	7.10E-01

	4U SNR=150 Frames				
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV
NMSE	2.28E-02	2.31E-02	1.99E-02	4.04E-02	3.88E-02
NLMSE	7.27E-01	6.96E-01	6.48E-01	8.90E-01	9.33E-01
NMSE	2.25E-02	2.32E-02	1.99E-02	3.66E-02	3.64E-02
NLMSE	7.28E-01	7.08E-01	6.56E-01	8.90E-01	9.29E-01
(hr2)					
NMSE	1.36E-02	1.01E-02	1.34E-02	3.96E-02	3.56E-02
NLMSE	5.48E-01	4.36E-01	5.30E-01	8.17E-01	9.18E-01
NMSE	1.34E-02	1.02E-02	1.37E-02	3.57E-02	3.44E-02
NLMSE	5.21E-01	4.14E-01	5.09E-01	7.98E-01	8.73E-01
(hr3)					
NMSE	3.30E-03	2.60E-03	4.10E-03	1.17E-02	8.60E-03
NLMSE	5.61E-01	4.85E-01	6.11E-01	7.81E-01	8.63E-01
NMSE	3.10E-03	2.50E-03	3.90E-03	7.70E-03	5.70E-03
NLMSE	5.38E-01	4.71E-01	5.84E-01	7.48E-01	7.30E-01

	4U SNR=75 Frames				
	RMMSE	MAP	POCS	MINT	MINT/IN
	2.39E-02	2.35E-02	2.59E-02	4.09E-02	3.98E-02
	7.40E-01	7.00E-01	7.43E-01	9.01E-01	9.51E-01
	2.35E-02	2.36E-02	2.58E-02	3.71E-02	3.76E-02
	7.41E-01	7.12E-01	7.54E-01	9.01E-01	9.47E-01
	1.54E-02	1.17E-02	1.98E-02	4.04E-02	3.74E-02
	5.82E-01	4.74E-01	6.49E-01	8.39E-01	9.61E-01
	1.53E-02	1.18E-02	2.01E-02	3.66E-02	3.64E-02
	5.56E-01	4.55E-01	6.26E-01	8.24E-01	9.19E-01
	4.10E-03	3.60E-03	6.40E-03	1.20E-02	9.40E-03
	6.23E-01	5.91E-01	7.84E-01	8.29E-01	9.39E-01
	3.90E-03	3.60E-03	6.10E-03	8.00E-03	6.50E-03
	6.00E-01	5.76E-01	7.59E-01	7.93E-01	8.02E-01

	8N SNR=inf Frames				
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV
NMSE	2.26E-02	2.24E-02	No Data	4.10E-02	3.79E-02
NLMSE	7.14E-01	6.84E-01		8.98E-01	9.27E-01
NMSE	2.13E-02	2.18E-02		3.72E-02	3.58E-02
NLMSE	7.10E-01	6.92E-01		8.98E-01	9.23E-01
(hr2)					
NMSE	1.36E-02	1.16E-02		4.03E-02	3.38E-02
NLMSE	5.27E-01	4.44E-01		8.34E-01	9.04E-01
NMSE	1.20E-02	9.53E-03		3.67E-02	3.30E-02
NLMSE	4.93E-01	4.08E-01		8.21E-01	8.66E-01
(hr3)					
NMSE	3.99E-03	5.27E-03		1.18E-02	8.16E-03
NLMSE	5.54E-01	5.79E-01		8.09E-01	8.64E-01
NMSE	2.70E-03	2.19E-03		7.78E-03	5.68E-03
NLMSE	5.04E-01	4.32E-01		7.76E-01	7.45E-01

	8N SNR=300 Frames				
	RMMSE	MAP	POCS	MINT	MINT/IN
	2.27E-02	2.25E-02	No Data	4.10E-02	3.79E-02
	7.15E-01	6.84E-01		8.97E-01	9.26E-01
	2.13E-02	2.18E-02		3.72E-02	3.58E-02
	7.11E-01	6.92E-01		8.97E-01	9.22E-01
	1.37E-02	1.17E-02		4.04E-02	3.38E-02
	5.30E-01	4.45E-01		8.35E-01	9.05E-01
	1.21E-02	9.60E-03		3.68E-02	3.31E-02
	4.97E-01	4.10E-01		8.23E-01	8.69E-01
	4.03E-03	5.34E-03		1.18E-02	8.20E-03
	5.58E-01	5.86E-01		8.09E-01	8.66E-01
	2.75E-03	2.23E-03		7.79E-03	5.73E-03
	5.07E-01	4.37E-01		7.76E-01	7.47E-01

	8N SNR=150 Frames				
(hr1)	RMMSE	MAP	POCS	MINT	MINT/INV
NMSE	2.28E-02	2.24E-02	No Data	4.11E-02	3.81E-02
NLMSE	7.16E-01	6.83E-01		9.00E-01	9.30E-01
NMSE	2.15E-02	2.19E-02		3.73E-02	3.61E-02
NLMSE	7.13E-01	6.92E-01		8.99E-01	9.26E-01
(hr2)					
NMSE	1.41E-02	1.19E-02		4.06E-02	3.44E-02
NLMSE	5.39E-01	4.48E-01		8.43E-01	9.16E-01
NMSE	1.26E-02	9.74E-03		3.69E-02	3.35E-02
NLMSE	5.05E-01	4.11E-01		8.27E-01	8.76E-01
(hr3)					
NMSE	4.19E-03	5.48E-03		1.19E-02	8.39E-03
NLMSE	5.72E-01	6.09E-01		8.22E-01	8.86E-01
NMSE	2.91E-03	2.44E-03		7.89E-03	5.91E-03
NLMSE	5.23E-01	4.65E-01		7.90E-01	7.67E-01

	8N SNR=75 Frames				
	RMMSE	MAP	POCS	MINT	MINT/IN
	2.38E-02	2.28E-02	No Data	4.16E-02	3.92E-02
	7.30E-01	6.85E-01		9.09E-01	9.46E-01
	2.24E-02	2.21E-02		3.78E-02	3.70E-02
	7.25E-01	6.92E-01		9.07E-01	9.41E-01
	1.58E-02	1.34E-02		4.13E-02	3.63E-02
	5.71E-01	4.83E-01		8.68E-01	9.64E-01
	1.41E-02	1.08E-02		3.77E-02	3.57E-02
	5.35E-01	4.35E-01		8.54E-01	9.26E-01
	4.72E-03	6.31E-03		1.22E-02	8.94E-03
	6.14E-01	6.84E-01		8.56E-01	9.38E-01
	3.45E-03	3.08E-03		8.14E-03	6.44E-03
	5.62E-01	5.20E-01		8.21E-01	8.18E-01

References

M.S. Alam, J.G. Bognar, R.C. Hardie, B.J. Yasuda, "High resolution infrared image reconstruction using multiple, randomly shifted, low resolution, aliased frames," SPIE Vol. 3063, 102-112, 1997.

H.C. Andrews and B.R. Hunt, *Digital Image Restoration*, Prentice-Hall, Englewood Cliffs NJ, 1977.

D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont MA, 1995.

S. Borman and R.L. Stevenson, "Super-Resolution from Image Sequences – A Review," 1998 Midwest Symposium on Circuits and Systems, IEEE Comput. Soc., 374-378, 1999.

P. Cheeseman, B. Kanefsky, R. Kraft, J. Stutz, and R. Hanson, "Super-Resolved Surface Reconstruction From Multiple Images," Technical Report FIA-94-12, NASA Ames Research Center, Artificial Intelligence Branch, 1994.

B. Cohen and I. Dinstein, "Resolution Enhancement By Polyphase Back-Projection Filtering," Proc. IEEE ICASSP, Vol. 5, 2921-2924, 1998.

B.R. Frieden, "Restoring with Maximum Likelihood and Maximum Entropy," J. Opt. Soc. America, Vol. 62 No. 4, 511-518, April 1972.

R.W. Gerchberg, "Super-resolution through error energy reduction," Optica Acta, Vol. 21 No. 9, 709-720, 1974.

J.C. Gillette, T.M. Stadtmiller, and R.C. Hardie, "Aliasing reduction in staring infrared imagers utilizing subpixel techniques," Optical Engineering, Vol. 34 No. 11, 3130-3137, November 1995.

J.W. Goodman, *Introduction to Fourier Optics*, McGraw-Hill, San Francisco, 1968.

R.C. Hardie, K.J. Barnard, and E.E. Armstrong, "Joint MAP Registration and High-Resolution Image Estimation Using a Sequence of Undersampled Images," IEEE Trans. Image Proc., Vol. 6 No. 12, 1621-1632, December 1997.

R.C. Hardie, K.J. Barnard, J.G. Bognar, and E.A. Watson, "High-resolution image reconstruction from a sequence of rotated and translated frames and its application to an infrared imaging system," Optical Engineering, Vol. 37 No. 1, 247-260, 1998.

J.L. Harris, "Diffraction and Resolving Power," J. Opt. Soc. America, Vol. 54 No. 7, 931-936, July 1964.

B.R. Hunt, "Super-Resolution of Images: Algorithms, Principles, Performance," Int. Journal. of Imaging Systems and Technology, Vol. 6, 297-304, 1995.

M. Irani and S. Peleg, "Super Resolution From Image Sequences," IEEE 10th Intl. Conference on Pattern Recognition, 115-120, 1990.

M. Irani and S. Peleg, "Improving Resolution by Image Registration," CVGIP: Graphical Models and Image Processing, Vol. 53 No. 3, 231-239, May 1991.

E. Kaltenbacher and R.C. Hardie, "High Resolution Infrared Image Reconstruction Using Multiple, Low Resolution, Aliased Frames," Proc. IEEE 1996 National Aerospace and Electronics Conference (NAECON), Vol. 2, 702-709, 1996.

J.S. Lim, *Two-Dimensional Signal and Image Processing*, Prentice Hall, Englewood Cliffs, NJ, 1990.

R.G. Lyon, J.M. Hollis, and J.E. Dorband, "A Maximum Entropy Method with a priori Maximum Likelihood Constraints," Astrophysical Journal, Vol. 478 No. 2, 658-662, April 1997.

A.V. Oppenheim and A.S. Willsky with S.H. Nawab, *Signals and Systems*, Second Edition, Prentice Hall, Upper Saddle River, NJ, 1997.

W.K. Pratt, *Digital Image Processing*, John Wiley & Sons, New York, 1978.

D.L. Ruderman and W. Bialek, "Seeing Beyond the Nyquist Limit," Neural Computation, Vol. 4, 682-690, 1992.

R.R. Schultz and R.L. Stevenson, "Extraction of High-Resolution Frames from Video Sequences," IEEE Trans. Image Proc., Vol. 5 No. 6, 996-1011, June 1996.

P.J. Sementilli, M.S. Nadar, and B.R. Hunt, "Empirical Evaluation of a Bound on Image Superresolution Performance," SPIE, Vol. 2302, 178-187, 1994.

M.I. Sezan and A.MTekalp, "Adaptive Image Restoration with Artifact Suppression Using the Theory of Convex Projections," IEEE Trans. Acoustics, Speech, and Signal Proc., Vol. 38 No. 1, 181-185, Jan 1990.

M.I. Sezan and A.M. Tekalp, "Survey of recent developments in digital image restoration," Opt. Eng. 29(5), 393-404, May 1990.

H. Shekarforoush and R. Chellappa, "Multi-channel Super-resolution for Image Sequences with Application to Airborne Video Data," Image and Multidimensional Digital Signal Processing 1998 Proceedings, 207-210, 1998.

H. Stark and P. Oskoui, "High-resolution Image Recovery from Image-plane Arrays, Using Convex Projections," J. Opt. Soc. of America A, Vol. 6 No. 11, 1715-1726, 1989.

R.Y. Tsai and T.S. Huang, "Multiframe image restoration and registration," *Advances in Computer Vision and Image Processing*, Eds. T.Y. Tsai and T.S. Huang, Vol. 1, 317-339, 1984.

E.A. Watson, R.A. Muse, and F.P. Blommel, "Aliasing and blurring in microscanned imagery," SPIE, Vol. 1689, 242-250, 1992.