# NONLINEAR INTERPOLATION OF QUINCUNX SUB-SAMPLED IMAGES

by

Deva Ramanan

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Engineering with Distinction

Spring 2000

# NONLINEAR INTERPOLATION OF QUINCUNX SUB-SAMPLED IMAGES

by

Deva Ramanan

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

## ABSTRACT

Image interpolation is an issue which has recently received great attention. The interpolation of images is a process which arises in many current applications, such as the Internet and high definition television, whether it be within an image zooming context or within a decompression phase of a data compression system. This thesis considers various interpolation schemes, particularly focussing on non-linear methods because of their inherent ability to preserve sharp edges and detail. Simulation results are presented comparing and demonstrating various schemes, offering a variety of performance levels at the expense of added computational complexity and memory.

# Chapter 1

# AN INTRODUCTION TO IMAGE INTERPOLATION

Digital image interpolation is an issue that has recently received great attention. With the continued development of data communication schemes for image transmission over the Internet and image/video coding for *high definition television* (HDTV), novel and efficient techniques for digital image down-sampling and image interpolation are more sought after than ever before. This thesis focuses on the more involved interpolation phase, an area which provides much room for development of new methods.

## 1.1 Digital Image Processing: Background Information

A digital image is essentially a table of numbers. A grayscale image is a straightforward (x,y) table where a given value represents the intensity, or brightness, at that (x,y) position. Color images are defined by 3 tables, each representing the intensity of the red, green, and blue hues [5]. Processing a digital image is accomplished by simply applying an algorithm to a table of numbers to yield a new table of numbers. This is commonly implemented in a raster scan fashion by passing an observation window over an image, processing the central pixel of the given current window. For example, for a $3 \times 3$ pixel-dimension observation window, there are a total of 9 pixels; the 8 neighboring samples, plus the central pixel to be processed (Fig. 1.1). We can generate an observation vector based on those nine pixels, and feed that vector into an algorithm, which will in turn produce a new value for the central pixel in the window. This window is passed along all areas of the image until

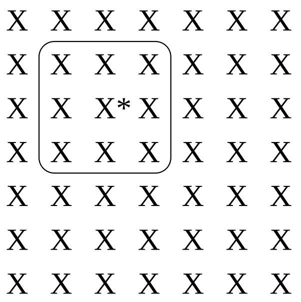


**Figure 1.1:** The $3 \times 3$ observation window encompassing X*, the current pixel to be filtered.

we have filtered all the pixels. Common applications of digital image processing are noise-filtering and edge enhancement. In noise-filtering, the filtering algorithm tries to identify certain image pixels which have a strikingly different intensity value from their neighbors, and processes them to yield a better fitting value. In edge enhancement, the filter attempts to find edges, or patterns of sharp intensity changes among neighboring pixels, and enhances them.

### 1.1.1 Frequency Domain Analysis

A digital image can also be interpreted in the frequency domain, as opposed to a spatial domain described by the table analogy. This is similar to viewing a auditory signal from the time domain (a one-dimensional domain, as opposed to the two dimensional spatial one) versus the frequency domain. Lower frequency signals manifest themselves in the spatial domain as regions of flat or gradual intensity changes, while the higher frequency signals manifest themselves in regions of sharp intensity changes. Thus high frequency components of an image include both noise pixels and edges. Noise-removing filters tend to reduce the high-frequency artifacts of an image, while edge-enhancing tends to accentuate the high-frequency artifacts of an image.

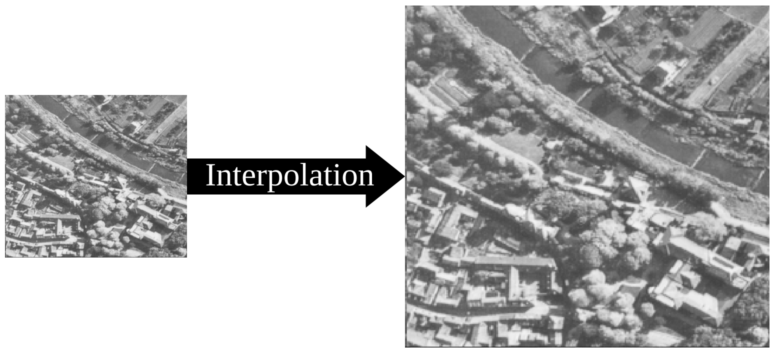


**Figure 1.2:** A generic interpolation scheme where a smaller image is enlarged, or interpolated.

## 1.2 Image Interpolation: An Extension of Image Filtering

Digital image interpolation can now be defined as an extension of digital image filtering. In any given interpolation scheme, a smaller image is the input, and a larger version is the desired output (Fig. 1.2). The input image can initially be filled or interlaced with zero pixels representing unknown values at specific locations to yeild the desired output-sized image. The specific locations to insert the zero pixels depends upon the method used to produce the smaller image; if it was created using a specific sub-sampling lattice, the zero pixels must be inserted in accordance with that lattice (Fig. 1.3). The zero-interlaced image can now be considered a noisy image, where the noise consists of the zero pixels that were just inserted. Thus the problem of image interpolation can essentially be reduced to the problem of noise filtering. However, since the exact location of the noisy, or zero, pixels are known, we only need to filter the effected pixel locations. This allows us in turn to be more aggressive in our filter algorithms since we know the pixels we are filtering are indeed noisy, and furthermore, we know which of the surrounding pixels are clean (i.e., were present in the original smaller image).

```
X X X X X      X   X   X      X 0 X 0 X
X X X X X                     0 0 0 0 0
X X X X X      X   X   X      X 0 X 0 X
X X X X X                     0 0 0 0 0
X X X X X      X   X   X      X 0 X 0 X
   (a)            (b)            (c)
```

**Figure 1.3:** A typical image interpolation progression based on the *Every Other Row and Column* (EORC) sub-sampling lattice. (a)Original image. (b)EORC sub-sampled image (one quarter the size of the original image).(c) Zero-Interlaced image (same size as original).

### 1.2.1 Traditional Noise-Reducing Filters

Treating interpolation as a noise-reducing problem, we can immediately apply traditional noise filters to process the zero pixels. Applying a traditional linear filter to the observation vectors generated by the filter windows exploits the spatial ordering of the neighboring pixels by outputting a weighted sum combination of them. However this method will tend to reduce most high-frequency signals of an image, producing blurred edges and smoothed details in the final interpolated image.

### 1.2.2 Median-Based Filters

Nonlinear filters, however, have proven more successful in the area of edge preservation. Zeng examined a series of median filters, in which the observation set includes four neighboring samples [14] of the central pixel to be interpolated. He further investigated extending the observation vector to include mean statistics based on the four samples, generally seeing improved results. However, median filtering itself can eliminate other high-frequency artifacts such as sharp corners and narrow lines because the rank-based filtering process neglects any spatial neighborhood information. We propose to couple both the rank and spatial ordering of the samples by using *weighted median* (WM) filters [2], which outputs a median based on spatially-assigned weights to the observation samples. Allowing the weights of the observation sample to be spatially determined allows us to output a weighted median where, for example, the center sample has a greater weight.

### 1.2.3 Permutation-Based Rank Selection Filters

We then consider *rank conditioned rank selection* (RCRS) and *extended permutation rank selection* (EPRS) filters, which fully integrate the rank and spatial orderings by using a permutation of the samples to map the spatial domain ordering to a rank domain one [6], [7], [3]. RCRS filters are a class of filters based upon the partitioning of an observation space using rank permutations of samples from the current filter window. A rank selection filtering operation is defined over each individual partition, allowing the RCRS filter to output one of the original samples from the filter window. The observation vector used to generate the permutations consist of the original N observation samples. EPRS filters use an extended observation vector that also includes K statistics based upon those N observation samples, similar to the extended vectors Zeng used [14]. This allows the rank selection operation of the EPRS filter to output not only one of the N original samples, but one of the K statistics as well. By selecting the appropriate N original samples and K statistics, both RCRS and EPRS filters can be applied to image interpolation.

## 1.3 Organization of Thesis

This remainder of this thesis is organized as follows. In Chapter 2, the *quincunx* sub-sampling lattice is presented. In Chapter 3, we consider methods for extracting and extending observation vectors from various-sized filter windows. We then apply these observation vectors to traditional linear and median filters in Chapter 4. In Chapter 5, we apply these observation vectors to the more complex permutation-based rank filters. We define an optimization procedure for the rank filters in Chapter 6. In Chapter 7, we present experimental results from the application of the quincunx-based interpolators.

# Chapter 2

# QUINCUNX SUB-SAMPLING LATTICE

This chapter introduces the quincunx sub-sampling lattice and outlines the basic interpolator structure associated with the lattice.

## 2.1 Transmission Channel

An important characteristic of any communication channel is the ability to transmit accurate images at a low bit rate. This is typically accomplished with the compression of the image data at the source end, a transmission of the data through a communication channel, and a decompression of the image at the receiver. One implementation of this system is using a sub-sampling/interpolation based coding/decoding procedure shown in Fig. 2.1. The coder and the decoder operations are a communication issue which falls outside the scope of this work. We will closely examine the sub-sampling and interpolation operations. There are many sub-sampling schemes possible; our research focused on the sub-sampling lattice. This lattice allows for a simpler interpolation phase needing only one interpolator structure, as opposed to the EORC lattice, in which multiple structures are needed to accomodate the three types of zero-pixels in the interlaced image (Fig. 1.3-c). This restriction to a single sub-sampling lattice allowed for a full examination of many interpolation schemes. However, our methods can be applied to other sub-sampling lattices as well, such as the EORC lattice.

The initial stage of the quincunx sub-sampling/interpolation progression can be seen in Fig. 2.2. The quincunx lattice reduces the number of samples by 2

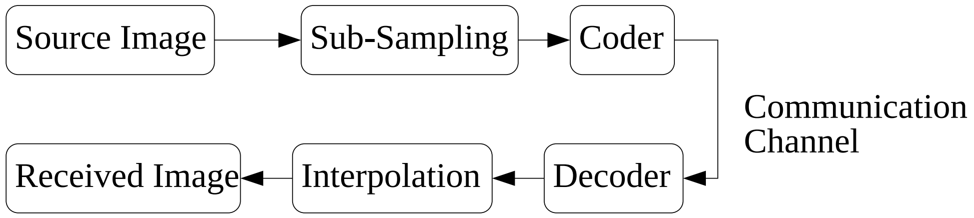


**Figure 2.1:** A typical image transmission channel implementing the sub-sampling/interpolation protocol.

```
X X X X X        X   X        0 X 0 X 0
X X X X X      X   X   X      X 0 X 0 X
X X X X X        X   X        0 X 0 X 0
X X X X X      X   X   X      X 0 X 0 X
X X X X X        X   X        0 X 0 X 0
   (a)            (b)            (c)
```

**Figure 2.2:** Sampling lattices. (a) Original. (b) Quincunx sub-sampled. (c) Quincunx zero-interlaced.

[14], [11]. It can subsequently be zero-interlaced to produce the original size image (Fig. 2.2-c) [2]. The next step in the interpolation progression is to filter the zero pixels in the zero-interlaced image, producing the completely reconstructed image. One preliminary point to consider is whether the original image should be low-pass filtered before it is subsampled to avoid any aliasing. We decided against this option since the low-pass filter would remove the high frequencies and sharp quality that we are trying to preserve through nonlinear interpolation. Additionally, many practical systems fail to include an antialiasing filtering operation. It should be noted that while eliminating this filtering step does preserve edges, it does introduce image distortions.

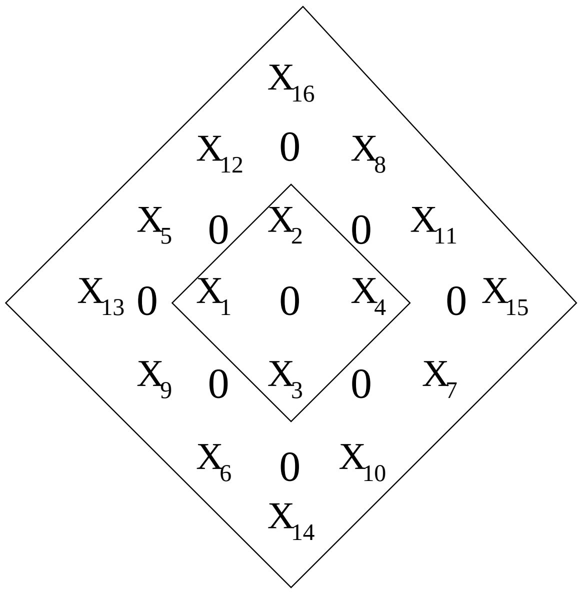


**Figure 2.3:** Interpolator structure for quincunx zero-interlaced image.

## 2.2 Interpolator Structure

The basic interpolator structure for the quincunx zero interlace is shown in Fig. 2.3. In this case, the central zero is the pixel to be filtered, or interpolated. The inner diamond encompasses the four adjacent neighboring samples, while the outer diamond encompasses the next set of known samples. Thus, using the given interpolator structure, there are up to 16 known samples that we can use to interpolate the value for the central zero pixel.

With the sub-sampling lattice and corresponding interpolator structure defined, the following chapter describes the procedure to extract observation vectors from the zero-interlaced image (Fig. 2.2-c) using the interpolator structure (Fig. 2.3).

# Chapter 3

# OBSERVATION VECTOR GENERATION

This chapter defines a set of observation vectors used for interpolation schemes based on the quincunx sub-sampling lattice and interpolator structure defined in the previous chapter.

We can represent the original image (Fig. 2.2-a) and the zero-interlaced version (Fig. 2.2-c) with the 2–dimensional discrete sequences $\{d(\mathbf{n})\}$ and $\{x(\mathbf{n})\}$, respectively. Note that from this point on, the index $\mathbf{n}$ is assumed and is used explicitly only when needed.

We will implement our filters in a raster-scan fashion, passing an observation window over the zero-interlaced image (Fig. 2.2-c), filtering only the zero pixel locations. At each zero pixel location, we must extract an observation vector consisting of the known surrounding samples.

## 3.1 Extracting the Observation Vector

Consider a 2–dimensional window function that spans $N$ original samples. Note we are defining the size of our window based on the number of original samples it encompasses, not including the zero pixels covered by the window. Thus even if we used all 16 neighboring samples from the interpolator structure shown in Fig. 2.3, we are defining our window size to be 16, rather than 25 (16 original and 9 zero pixels). For the central zero pixel being filtered, we consider the surrounding non-zero samples to create an observation window of $N$ samples, defining a corresponding observation vector of

$$\mathbf{x}_N = [x_1, x_2, \ldots, x_N]. \tag{3.1}$$

Using the interpolator structure from Fig. 2.3, we can extract the observation vector $\mathbf{x}_{16} = [x_1, x_2, \ldots, x_{16}]$ to create a 16 element filter window.

We can pass the observation vector $\mathbf{x}_N$ through any filter, using the output as the value for the center zero pixel. A typical linear scheme would exploit the spatial ordering of the observed samples by outputting a weighted average of $\mathbf{x}_N$. Increasing the computation and complexity, we could exploit the rank ordering of the samples by choosing the median of $\mathbf{x}_N$. We will, however, propose schemes combining both the rank and spatial orderings of the samples.

## 3.2 Extending the Observation Vector

Additional information about the window can be extracted by extending the observation vector to include $K$ characteristic statistics. Thus, we define an extended observation vector as

$$\begin{aligned} \tilde{\mathbf{x}}_{N,K} &= [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{N+K}], \qquad (3.2) \\ &= [x_1, x_2, \ldots, x_N, F_1(\mathbf{x}_N), F_2(\mathbf{x}_N), \ldots, F_K(\mathbf{x}_N)]. \end{aligned}$$

where $F_i(\mathbf{x}_N)$ is some function of the original observation vector. Note that the standard observation vector $\mathbf{x}_N$ is simply a special case of $\tilde{\mathbf{x}}_{N,K}$, for $K = 0$.

## 3.3 Implemented Observation Vectors

To examine the effects of $N$ and $K$ on the performance of various filters, define two sets of observation vectors. In the initial set, consider the original observation vector for $K = 0$. Define 9 observation vectors for N = 2,3,...,8,12,16.

$$\mathbf{x}_2 = [x_1, x_2]. \tag{3.3}$$

$$\mathbf{x}_3 = [x_1, x_2, x_3]. \tag{3.4}$$

$$\mathbf{x}_4 = [x_1, x_2, x_3, x_4]. \tag{3.5}$$

$$\mathbf{x}_5 = [x_1, x_2, x_3, x_4, x_5]. \tag{3.6}$$

$$\mathbf{x}_6 = [x_1, x_2, x_3, x_4, x_5, x_6]. \tag{3.7}$$

$$\mathbf{x}_7 = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]. \tag{3.8}$$

$$\mathbf{x}_8 = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]. \tag{3.9}$$

$$\mathbf{x}_{12} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}]. \tag{3.10}$$

$$\mathbf{x}_{16} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}] \tag{3.11}$$

To examine the effect of adding statistics, define a set of extended observation vectors based on the original vectors above. Noting the success of adding an overall mean to EPRS filters as an edge enhancement mechanism [7], we will also extend our filters with linear averaging statistics as well. However, to derive increased statistical information about our observation window, we will add directional means of samples positioned along lines which pass through the center sample. For example, for $\mathbf{x}_4$, we will extend the vector by two elements; the average of $x_1$ and $x_4$, and the average of $x_2$ and $x_3$. Note from Fig. 3.1-a that $x_1$ and $x_4$ are positioned along a horizontal line crossing the center pixel, and $x_2$ and $x_3$ are positioned along a similar vertical line. Since our directional mean statistics are exploiting the symmetry of the original observation window, we will only extend the observation windows which are inherently symmetric. Thus, for the vectors $\mathbf{x}_4, \mathbf{x}_8, \mathbf{x}_{12}$, and $\mathbf{x}_{16}$, we will add analogous directional mean statistics (Fig. 3.1) to define the corresponding extended vectors

$$\tilde{\mathbf{x}}_{4,2} = [\mathbf{x}_4, \bar{x}_{\{1,4\}}, \bar{x}_{\{2,3\}}], \tag{3.12}$$

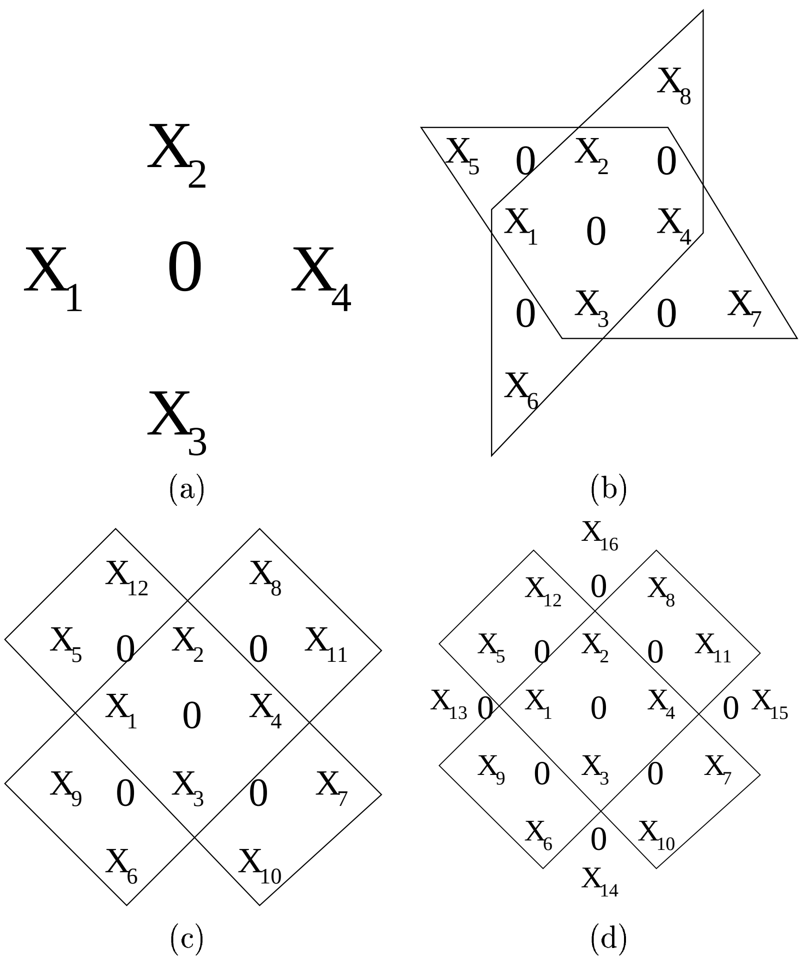
$$\tilde{\mathbf{x}}_{8,2} = [\mathbf{x}_8, \bar{x}_{\{1,2,3,4,5,7\}}, \bar{x}_{\{1,2,3,4,6,8\}}], \tag{3.13}$$

$$\tilde{\mathbf{x}}_{12,2} = [\mathbf{x}_{12}, \bar{x}_{\{1,2,3,4,5,7,10,12\}}, \bar{x}_{\{1,2,3,4,6,8,9,11\}}], \tag{3.14}$$

$$\tilde{\mathbf{x}}_{16,4} = [\mathbf{x}_{16}, \bar{x}_{\{1,2,3,4,5,7,10,12\}}, \bar{x}_{\{1,2,3,4,6,8,9,11\}}, \bar{x}_{\{2,3,14,16\}}, \bar{x}_{\{1,4,13,15\}}], \tag{3.15}$$

where $\bar{x}_{\{i1,i2,\ldots,in\}} = \text{mean}(x_{i1}, x_{i2}, \ldots, x_{in})$. Note that because of the added symmetry in $\mathbf{x}_{16}$, we were able to directly extend the vector by four averaging statistics denoting the four main directional lines passing through the center pixel (Fig. 3.1-c).

We can now perform our filtering operation directly on the defined observation vectors. For example, in a simple linear or median scheme, we output the mean or median of the vector, respectively. The next two chapters define a series of filters which operate on one or both sets of vectors defined above. Since the original observation vector $\mathbf{x}_N$ is simply a special case of the extended observation vector $\tilde{\mathbf{x}}_{N,K}$ for $K = 0$, filters used for both vectors are only explicitly defined for the extended vector $\tilde{\mathbf{x}}_{N,K}$.



$X_2$ $X_1$ 0 $X_4$ $X_3$
(a)
$X_8$ $X_5$ 0 $X_2$ 0 $X_1$ 0 $X_4$ 0 $X_3$ 0 $X_7$ $X_6$
(b)
$X_{12}$ $X_8$ $X_5$ 0 $X_2$ 0 $X_{11}$ $X_1$ 0 $X_4$ $X_9$ 0 $X_3$ 0 $X_7$ $X_6$ $X_{10}$
(c)
$X_{16}$ $X_{12}$ 0 $X_8$ $X_5$ 0 $X_2$ 0 $X_{11}$ $X_{13}$ 0 $X_1$ 0 $X_4$ 0 $X_{15}$ $X_9$ 0 $X_3$ 0 $X_7$ $X_6$ 0 $X_{10}$ $X_{14}$
(d)

**Figure 3.1:** Interpolator structures with directional mean statistics graphically denoted for (a)$\mathbf{x}_{4,2}$, (b)$\mathbf{x}_{8,2}$, (c)$\mathbf{x}_{12,2}$, and (d)$\mathbf{x}_{16,4}$.

# Chapter 4

# TRADITIONAL FILTERS

In this chapter, we describe traditional filters for the various observation vectors defined in the previous chapter. Prior to the filter definitions, we first outline our criteria of evaluating their performance.

## 4.1 Error Criteria

Since some filters are designed to be optimized over a certain error criteria, we will explicitly define the *Mean Squared Error* (MSE) and the *Mean Absolute Error* (MAE) between the original $\{d(\mathbf{n})\}$ and reconstructed image, denoted $\{\hat{d}(\mathbf{n})\}$. Here, our reconstructed image is our original zero-interlaced image $\{x(\mathbf{n})\}$ after the zero-pixels have been filtered.

During the raster-scan progression of the filter window, a sequence of observation vectors will be encountered for each zero pixel filtered. We can denote this sequence $\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_1), \tilde{\mathbf{x}}_{N,K}(\mathbf{n}_2), \ldots, \tilde{\mathbf{x}}_{N,K}(\mathbf{n}_\lambda)$, where $\lambda$ is equal to the number of zero pixels in the zero-interlaced image $\{x(\mathbf{n})\}$ to be filtered. Similarly, we can write the corresponding desired values and filtered estimates as the sequences $d(\mathbf{n}_1), d(\mathbf{n}_2), \ldots, d(\mathbf{n}_\lambda)$ and $F(\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_1)), F(\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_2)), \ldots, F(\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_\lambda))$ respectively. We can now characterize the sample MAE of the reconstructed image as

$$MAE = \frac{1}{\lambda}\sum_{i=1}^{\lambda} \mid d(\mathbf{n}_i) - \hat{d}(\mathbf{n}_i) \mid, \tag{4.1}$$

$$= \frac{1}{\lambda}\sum_{i=1}^{\lambda} | d(\mathbf{n}_i) - F(\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_i)) |. \quad (4.2)$$

The corresponding sample MSE between the original and reconstructed image is

$$MSE = \frac{1}{\lambda}\sum_{i=1}^{\lambda} | d(\mathbf{n}_i) - F(\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_i)) |^2. \quad (4.3)$$

Note that both these values are averaged only over the filtered pixels, not the entire image (ie, the original pixels preserved during the sub-sampling are not included).

## 4.2 Linear Filters

Consider a distribution analysis of the pixel samples in the observation vector. In a purely randomly generated image, the distribution can be modeled by the standard Gaussian distribution

$$f(x) = ce^{(|x-\beta|/\sigma)^2}, \quad (4.4)$$

where $\beta$ is the mean and $\sigma$ is the standard deviation (Fig. 4.1-a). It can be shown that the Maximum Likelihood(ML) estimate of location of the distribution is the sample mean [1]. Thus if we interpret our filtered pixel as the ML estimate of the observation window, we can define our first linear filter Lin1 to be the straightforward mean.

$$F_{Lin1}(\mathbf{x}_N) = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad (4.5)$$

where $x_1, x_2, \ldots, x_N$ are the samples from the observation vector. Note in this section we are only defining our linear filters for the original observation vector $\mathbf{x}_N$, since extended the vector by linear statistics provides no performance gains for linear filters.

We can extend the filter to a broader class if allow the observation samples to be independent, but not identically distributed. Consider the generalized Gaussian distribution where the samples still have a common location parameter $\beta$, but each $x_i$

has a unique standard deviation $\sigma_i$. The samples spatially closer to the central zero pixel would have a smaller associated deviation from the ML estimate. These unique $\sigma$ values manifest themselves in the filter as weights for the associated observation samples when calculating the weighted mean. Thus we can define our next linear filter to be

$$F_{Lin2}(\mathbf{x}_N) = \frac{\sum_{i=1}^{N} w_i \cdot x_i}{\sum_{i=1}^{N} w_i} \tag{4.6}$$

where $w_i$ can be set according to the particular application. Specifically, for $F_{Lin2}$, they will be set to the inverse of the Euclidean distance from center zero pixel to $x_i$.

An optimal weight vector $\mathbf{w}^*$ can be calculated using the Weiner filter [8]. This filtered method optimizes linear weights under the MSE statistical criterion. In this case, the filter output is given by

$$\begin{aligned} F_{Lin3}(\mathbf{x}_N) &= \frac{\sum_{i=1}^{N} w_i^* \cdot x_i}{\sum_{i=1}^{N} w_i^*}, & (4.7) \\ &= \mathbf{w}^* * \mathbf{x}_N^T. & (4.8) \end{aligned}$$

### 4.2.1 Optimization of Weiner Coefficients

Substituting the matrix notation form in Eq. 4.8 into the MSE definition from Eq. 4.3, we can we can define the error of the filtered estimate for this observation vector as

$$\begin{aligned} MSE &= \frac{1}{\lambda}\sum_{i=1}^{\lambda} \mid d(\mathbf{n}_i) - F_{Lin3}(\mathbf{x}_N(\mathbf{n}_i)) \mid^2, & (4.9) \\ &= \frac{1}{\lambda}\sum_{i=1}^{\lambda} \mid d(\mathbf{n}_i) - \mathbf{w}^* * \mathbf{x}_{N(\mathbf{n}_i)}^T \mid^2. & (4.10) \end{aligned}$$

We can redefine the MSE in terms of the expected value operator $E\{\}$ from standard theory by considering $\mathbf{x}_N(\mathbf{n}_i)$ and $d(\mathbf{n}_i)$ to be specific values of random variables $\mathbf{x}_N(\mathbf{n})$ and $d(\mathbf{n})$. Assuming the random variables are Gaussian, the expected value reduces to the mean [12]. We can now write the MSE as a function of the weight vector $\mathbf{w}^*$ to yeild

$$MSE(\mathbf{w}^*) \quad = \quad E\{[d(\mathbf{n}) - \mathbf{w}^* * \mathbf{x}_N(\mathbf{n})^T]^2\}. \tag{4.11}$$

The minimum $MSE$ can be derived by differentiating with respect to $\mathbf{w}^*$ and setting the the resulting equation to 0 to yeild

$$\mathbf{w}^* = \mathbf{R}_{xx}^{-1} * \mathbf{P}_{xd}, \tag{4.12}$$

where

$$\mathbf{R}_{xx} \quad = \quad E\{\mathbf{x}_N(\mathbf{n}) * \mathbf{x}_N(\mathbf{n})^T\}, \tag{4.13}$$

$$\mathbf{P}_{xd} \quad = \quad E\{\mathbf{x}_N(\mathbf{n}) * d(\mathbf{n})\}. \tag{4.14}$$

The auto-correlation of $\mathbf{x}_N(\mathbf{n})$, $\mathbf{R}_{xx}$, and the cross-correlation between $\mathbf{x}_N(\mathbf{n})$ and $d(\mathbf{n})$, $\mathbf{P}_{xd}$, are statistics which can be summed over all the $\lambda$ observation vectors encountered in the training image.

## 4.3 Median Filters

The Gaussian distribution, however, is often not an accurate model for a image. Such an image would tend to have blurred edges, no sharp detail and smoothed over lines; the very cues that the human visual system gravitates toward when looking at an image. Thus, for our application of image interpolation, we want to maintain the sharp edges and details.

In that case, a better distribution model for the image would account for sharp changes in pixel values (representing edges and sharp detail) rather than smooth gradual intensity changes. If we generalize Eq. 4.4 to

$$f(x) = ce^{(|x-\beta|/\sigma)^p}, \tag{4.15}$$

we can model a more impulsive sample distribution be decreasing $p$ (Fig. 4.1). In fact, the function evaluated at $p = 1$ is well known as the Laplacian distribution function.

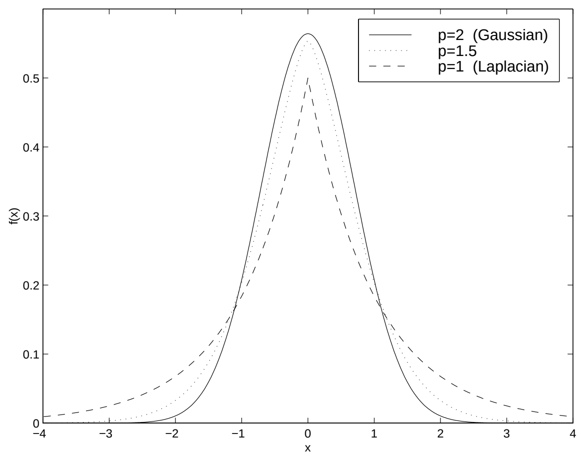


**Figure 4.1:** The generalized Gaussian distribution for $p = 2$ (standard Gaussian distribution), $p = 1.5$, and $p = 1$ (Laplacian distribution).

The ML estimator for Laplacian distribution is the sample median [1]. Hence we will define our nonlinear filter to be the median value of the observation vector

$$F_{Med}(\tilde{\mathbf{x}}_{N,K}) = \text{MED}[\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{N+K}]. \tag{4.16}$$

Note that for our median-based filters, we are considering extending the observation vectors by $K$ linear statistics to determine if a performance gain results. The median filter, as well as all of the nonlinear filters we will define, require a sorting operation which is more computationally taxing than calculating a simple linear weighted combination. However, novel methods for parallel and pipelined sorting algorithms are becoming more prevalent and will reduce the running times needed for sorting [15].

We can analogously extend our median filter into a generic class by allowing the individual observation samples to have a spatially dependent standard deviation from the center pixel [13]. We can now define a weighted median (WM) filter as

$$F_{WM}(\tilde{\mathbf{x}}_{N,K}) = \text{MED}[w_1 \diamond \tilde{x}_1, w_2 \diamond \tilde{x}_2, \cdots, w_{N+K} \diamond x_{N+K}], \tag{4.17}$$

where $\diamond$ is the replication operator defined as $w_i \diamond x_i = \overbrace{x_i, x_i, \cdots, x_i}^{w_i \text{ times}}$. Note that these weights often yield the best results when they are identical to the linear filter weights; the reason for emphasizing certain samples in determining filter output is often independent of the filtering scheme. We thus define two WM filters, $F_{WM1}$ and $F_{WM2}$ which use the distance-derived and linear-optimized weights respectively (i.e., $F_{WM1}$ uses weights identical to $F_{Lin2}$, and $F_{WM2}$ uses weights identical to $F_{Lin3}$).

Recall, however, linear weights were only defined for the original observation vector $\mathbf{x}_N$. Since the $K$ additional statistics are all linear averages of select samples from the vector, we define the associated weights as the inverse of the mean distance of the averaged samples to the center pixel. Thus, a statistic averaging samples close to the center pixel would be weighed more than a statistic averaging samples further away. These position-based weights are only applicable to $F_{WM1}$, whose original sample weights are also position-based. $F_{WM2}$, based on the linear Weiner statistics, cannot be applied on the extended observation vectors.

## 4.4 Passband Analysis

The passband signals of a filter are the signals which pass through the filter unaltered. Passband analysis is a useful tool in determining what the characteristic behavior of a filter is, i.e., what types of signals will and will not go through.

### 4.4.1 Linear Filters: Frequency Domain Analysis

The passband of linear filters can easily be characterized by a frequency domain analysis. All linear filters have one or more passband frequency ranges, and any input signal within the passband will pass through completely unaltered, while any input signal outside the passband will be attenuated or will not pass through at all. Thus, using linear filters, we can isolate the frequencies of signals we wish to retain in our final image. Lin1 and Lin2 both behave like low-pass filters, implying mostly low frequencies will be retained in the filtered image. The low frequency signals manifest themselves in the regions of solid or slightly changing intensity values. However, as previously stated, we are interested in preserving the regions of sharp intensity changes where edges and small detail occur. These regions are characterized by a combination of signals of widely varying high and low frequencies. These are hence not preserved in a linear filtering operation.

### 4.4.2 Median Filters: Root signal analysis

Frequency analysis is based on the superposition property which only applies to linear filters [3]. Therefore, a new mechanism for characterizing passband signals of nonlinear filters must be defined. For our purposes, we will define a mechanism for characterizing median-based nonlinear filters.

We can define a root image as any image which does not change when median-filtered. The median filter has a corresponding root set, or set of input signals which are invariant under the filtering operation. By examining the root set, we can see the types of image regions which will be preserved under the filtering operation. It can be shown that the root set consists of signals with constant neighborhoods and edges only [3]. Here, we are defining a constant neighborhood as any solid region of constant intensity value which could occupy the majority of the observation window. Thus, for a 3 X 3 filter window, a solid 2 X 2 or bigger region of constant value would be considered a constant neighborhood. Similarly, an edge is defined as a monotonic region between two constant neighborhoods of different value, where the monotonic region itself cannot contain any constant neighborhoods. Applying the median operation to both such regions does not change the input image. Hence, we immediately see one of the benefits of using a median-based filter, i.e., the ability to completely preserve edges.

## 4.5 Summary of traditional filters

The linear filters $F_{Lin1}, F_{Lin2}$, and $F_{Lin3}$ exploit the spatial ordering of the pixels in the observation window by outputting a weighted sum combination of them. The basic median filter $F_{Med}$ exploits the rank ordering of the pixels by outputting the mean, while the WM filters $F_{WM1}$ and $F_{WM2}$ partially couple the rank and spatial ordering by outputting a weighted median.

The next two chapters define $F_{EPRS}$, which fully integrates the the spatial ordering of the observation window pixels to their rank ordering.

# Chapter 5

# RANK INTERPOLATORS

All of the defined observation vectors and extended observation vectors can be used as input vectors for more complex filters as well as the simpler linear and median-based filters. In this chapter, we will introduce the RCRS and EPRS filters, which both fully use the rank and spatial orderings of the samples by considering a permutation of the samples from the spatial ordering to the rank ordering. The RCRS filter is applied on the standard observation vector, while the EPRS filter is applied on the extended observation vector. However, since $\mathbf{x}_N$ is simply a special case of $\tilde{\mathbf{x}}_{N,K}$, for $K = 0$, the RCRS filter is a limiting case of the EPRS filter using an extended observation vector of $\tilde{\mathbf{x}}_{N,0}$. For the remainder of this chapter, we will only explicitly define the EPRS filter.

## 5.1 Creating the Rank Feature Vector

We will collectively refer to all the samples from our extended observation vector as the sample pool. To create a rank ordering of the sample pool, sort $\tilde{\mathbf{x}}_{N,K}$ by rank to define

$$\tilde{x}_{(1)} \leq \tilde{x}_{(2)} \leq \cdots \leq \tilde{x}_{(N+K)}. \tag{5.1}$$

The ranks of the $N$ samples and $K$ statistics can be related to their spatial position (within the observation vector) by defining $r_i$ to be the rank of the sample at index $i$ of $\tilde{\mathbf{x}}_{N,K}$. This allows for the creation of the rank pool vector

$$\mathbf{r} = [r_1, r_2, \ldots, r_{N+K}], \tag{5.2}$$

which contains the ranks of the $N$ samples and $K$ statistics. Note that the rank pool vector can be used to describe the equivalence $\tilde{x}_i \equiv \tilde{x}_{(r_i)}$.

Due to the factorial growth in the number of permutations as the filter window size increases, we are often not able to practically use the full permutation contained in the rank pool vector [3]. Thus, if we extract only the ranks of select locations from the sample pool, we can reduce the number of permutations we need to consider and index.

Hence, instead of including all N original samples and K statistics in our final rank feature vector, include M original samples and L statistics, such that $0 \leq M \leq N$ and $0 \leq L \leq K$. Typically we will select the closest $M$ samples from our sample pool along with all of our statistics (such that $L = K$) to be included in the final rank feature vector. We can now define the rank feature vector to be

$$\mathbf{r}^* = [r_{\gamma_1}, r_{\gamma_2}, \ldots, r_{\gamma_M}, r_{\phi_1}, r_{\phi_2}, \ldots, r_{\phi_L}] \in \Omega_{\alpha,\beta}, \tag{5.3}$$

where $1 \leq \gamma_i \leq N$, $N+1 \leq \phi_i \leq N+K$. The rank feature vector permutation space (denoted $\Omega_{\alpha,\beta}$, where $\alpha = N+K$ and $\beta = M+L$) consists of all combinations of ranks that could possibly occur. A further examination of the partitioning of this permutation space is given in the following optimization chapter.

This rank vector can be used as an input into a selection rule $\mathcal{S}(\cdot)$, which will produce desired rank to output [6]. This allows the center zero pixel to take on any value in $\tilde{\mathbf{x}}_{N,K}$, which consists of the original samples and statistics.

## 5.2 Parity of the Extended Observation Vector

We address the issue of the number of elements in the extended observation vector. Median filters, when faced with an even-numbered $\tilde{\mathbf{x}}_{N,K}$ (with no true median value), will output the average of the two center-ranked samples. This behavior proves to be very desirable in many cases. To allow EPRS filters the ability to do the same, define

$$\hat{\mathbf{x}} = \begin{cases} \tilde{\mathbf{x}}_{N,K} & \text{if } N+K \text{ is odd} \\ [\tilde{\mathbf{x}}_{N,K},\ \text{median}(\tilde{\mathbf{x}}_{N,K})] & \text{if } N+K \text{ is even} \end{cases} \tag{5.4}$$

The vector $\hat{\mathbf{x}}$ is identical to $\tilde{\mathbf{x}}_{N,K}$, except a median value is appended when $\tilde{\mathbf{x}}_{N,K}$ has an even number of elements.

## 5.3 Completed filter definition

Defining $\hat{x}_{(i)}$ as the $i^{th}$ rank-ordered element of $\hat{\mathbf{x}}$, the output of filter can now be defined as

$$F_{EPRS}(\tilde{\mathbf{x}}_{N,K}) = \hat{x}_{(\mathcal{S}(\mathbf{r}^*))}, \tag{5.5}$$

where $\mathcal{S} : \Omega_{\alpha,\beta} \mapsto \{1, 2, \ldots, N\}$. The EPRS selection filter rule $\mathcal{S}(\cdot)$ can be optimized under the least MAE or MSE given the training sequences $\{d(\mathbf{n})\}$ and $\{x(\mathbf{n})\}$. A detailed definition and optimization procedure for $\mathcal{S}(\cdot)$ is presented in the next chapter.

# Chapter 6

# OPTIMIZATION OF EPRS FILTERS

This chapter describes the basic optimization procedure for EPRS filters. We can optimize under the MAE or MSE criterion. We will only explicitly define the optimization procedure for the MAE, but the MSE follows an analogous procedure.

Minimizing the mean error as a function of the selection rule $\mathcal{S}(\cdot)$ is analogous to minimizing the total error as a function of $\mathcal{S}(\cdot)$. Hence we will begin by defining the *total absolute error* (TAE) as

$$TAE = \sum_{i=1}^{\lambda} \mid d(\mathbf{n}_i) - F_{EPRS}(\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_i)) \mid = \sum_{i=1}^{\lambda} \mid d(\mathbf{n}_i) - \hat{x}_{(\mathcal{S}(\mathbf{r}^*(\mathbf{n}_i)))} \mid . \tag{6.1}$$

from Eq. 4.2. The selection rule $\mathcal{S}(\cdot)$ which minimizes Eq. 6.1 is referred to as the optimal selection rule $\mathcal{S}_{opt}(\cdot)$. In order to define $\mathcal{S}_{opt}(\cdot)$, the rank feature vectors comprising the feature space $\Omega_{\alpha,\beta}$ must be indexed.

## 6.1 Permutation Indexing Method

This section provides a theoretical and practical examination of a novel method to efficiently index the feature vector partitions of the observation space $\Omega_{\alpha,\beta}$ used in the rank filters. Kreher outlines a similar method for indexing subsets [10], which we extend to index permutations. Let $\alpha = N + K$, representing the total number of elements in our original pool, and let $\beta = M + L$, representing the number of elements we wish to consider. The challenge now becomes to efficiently index all the possible permutations of $\alpha$ elements taken $\beta$ at a time, denoted

${}_{\alpha}\mathbf{P}_{\beta} = \frac{\alpha!}{(\alpha-\beta)!}$. These permutations represent all the feature vectors which comprise the feature space $\Omega_{\alpha,\beta}$, with cardinality $|\Omega_{\alpha,\beta}| =_{\alpha} \mathbf{P}_{\beta}$.

Define the $\beta$ rank values which comprise a feature vector $\mathbf{r}^*$ as

$$\mathbf{r}^* = [r_1^*, r_2^*, \ldots, r_\beta^*], \tag{6.2}$$

where $1 \leq r_i^* \leq \alpha$. Consider the natural lexographic ordering of the feature vector permutations where the lowest available number for the $i$th vector position is used, starting from $i = 0$. Such an ordering is shown for the feature space $\Omega_{5,3}$ in Table 6.1.

To construct an indexing algorithm, we essentially need to count the number of permutations that precede a given vector $\mathbf{r}^*$ in the above ordering. We first note that there are ${}_{\alpha-1}\mathbf{P}_{\beta-1}$ permutations $\mathbf{R} \in \Omega_{\alpha,\beta}$ such that $R_1 = r_1^*$, where $\mathbf{R} = [R_1, R_2, \ldots, R_\beta]$. We can generally write for any $i \leq \beta$ integers $r_1^*, r_2^*, \ldots, r_i^*$, there are ${}_{\alpha-i}\mathbf{P}_{\beta-i}$ permutations $\mathbf{R}$ such that $R_1 = r_1^*, R_2 = r_2^*, \ldots, R_i = r_i^*$.

Thus for a given permutation vector $\mathbf{r}^* = [r_1^*, r_2^*, \ldots, r_\beta^*]$ we can count all the permutations preceding it in the ordering as

- The permutations $\mathbf{R}$ with $R_1 < r_1^*$.
- The permutations $\mathbf{R}$ with $R_1 = r_1^*$ and $R_2 < r_2^*$.
- The permutations $\mathbf{R}$ with $R_1 = r_1^*$,$R_2 = r_2^*$ and $R_3 < r_3^*$ .

$\vdots$

- The permutations $\mathbf{R}$ with $R_1 = r_1^*$,$R_2 = r_2^*$,$\ldots$,$R_\beta < r_\beta^*$.

Note that as we traverse the $\mathbf{r}^*$ vector, we must know how many possible numbers $R_i < r_i^*$ which could occupy the $i$th position in the permutation vector. Following our next-lowest number ordering, the number of possible values for the $i$th slot is $r_i^*$ minus the number of values less than $r_i^*$ we have already using previously in the vector (for $r_1^*, r_2^*, \ldots, r_{i-1}^*$). We can write this as a coefficient

$$Bi = r_i^* - \sum_{j=1}^{i-1} I(r_j^* < r_i^*) \tag{6.3}$$

| $\mathbf{r}^*$ | $Index(\mathbf{r}^*)$ |
|---|---|
| [1,2,3] | 0 |
| [1,2,4] | 1 |
| [1,2,5] | 2 |
| [1,3,2] | 3 |
| [1,3,4] | 4 |
| [1,3,5] | 5 |
| [1,4,2] | 6 |
| [1,4,3] | 7 |
| [1,4,5] | 8 |
| [1,5,2] | 9 |
| [1,5,3] | 10 |
| [1,5,4] | 11 |
| [2,1,3] | 12 |
| [2,1,4] | 13 |
| [2,1,5] | 14 |
| [2,3,1] | 15 |
| [2,3,4] | 16 |
| [2,3,5] | 17 |
| [2,4,1] | 18 |
| [2,4,3] | 19 |
| [2,4,5] | 20 |
| [2,5,1] | 21 |
| [2,5,3] | 22 |
| [2,5,4] | 23 |
| [3,1,2] | 24 |
| $\vdots$ | $\vdots$ |

**Table 6.1:** Example feature vector indexing for feature space $\Omega_{5,3}$.

for $i = 1, 2, \ldots, \beta$.

The function $I(\cdot)$ in 6.3 is the indicator function defined by

$$\mathrm{I}(event) = \begin{cases} 1 & \text{if event is true} \\ 0 & \text{if event is false} \end{cases} \tag{6.4}$$

Using this counting method, we can write the final indexing algorithm as

$$Index(\mathbf{r}^*) = \sum_{i=1}^{\beta} (B_i)({}_{\alpha-i}\mathbf{P}_{\beta-i}) \tag{6.5}$$

## 6.2 Defining $\mathcal{S}_{opt}(\cdot)$

Now that a method for indexing the feature vectors has been described, the entire feature space $\Omega_{\alpha,\beta}$ can be expressed as the set of indexed partitions

$$\Omega_{\alpha,\beta} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{|\Omega_{\alpha,\beta}|}\}. \tag{6.6}$$

We can express the sequence of rank feature vectors encountered as the observation window passes through $\{x(\mathbf{n})\}$ as $r^*(\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_1)), r^*(\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_2)), \ldots, r^*(\tilde{\mathbf{x}}_{N,K}(\mathbf{n}_\lambda))$. We will henceforth represent the sequence with the simpler notation $r^*(\mathbf{n}_1), r^*(\mathbf{n}_2), \ldots,$ $r^*(\mathbf{n}_\lambda)$. Let $\Psi_i = Index(\mathbf{r}^*(\mathbf{n}_i))$, yielding the equivalence relation $\mathbf{r}_{\Psi_i} = \mathbf{r}^*(\mathbf{n}_i)$. We can now define $\Gamma_{j,\lambda} = \{i \in \{1, 2, \ldots, \lambda\} : \Psi_i = j\}$, which represents the set of all feature vectors encountered in the zero-interlaced image $\{x(\mathbf{n})\}$ with index $j$ (following the defined indexing algorithm).

The total error incurred by selecting the $k^{th}$ order statistic for all the instances where rank vector $\mathbf{r}_j$ is observed in the zero-interlaced image is written as

$$\mathcal{E}_j(k) = \sum_{i \in \Gamma_{j,\lambda}} |\ d(\mathbf{n}_i) - \hat{x}_{(k)}(\mathbf{n}_i)\ |. \tag{6.7}$$

where $1 \leq k \leq \alpha$ and $j = 1, 2, \ldots, |\Omega_{\alpha,\beta}|$. Rewriting the total error from Eq. 6.1 as a sum of errors partitioned according to the feature vector produces

$$TAE = \sum_{j=1}^{|\Omega_{\alpha,\beta}|} \mathcal{E}_j(\mathcal{S}(\mathbf{r}_j)). \tag{6.8}$$

Realizing that the total error (Eq. 6.8) will be minimized if and only if the individual partition errors (Eq. 6.7) are minimized, we can now define the optimal selection rule as

$$\mathcal{S}_{opt}(\mathbf{r}_j) = k \ : \ \mathcal{E}_j(k) \leq \mathcal{E}_j(l) \ \forall \ l \neq k \tag{6.9}$$

for $j = 1, 2, \ldots, |\Omega_{\alpha,\beta}|$. For each rank vector partition in the feature space, the optimal selection rule will select the the order statistic which will produces the lowest error for that vector partition.

With this chapter, we culminate the definition of $F_{EPRS}$, as well as the definitions of all the filters used in our interpolation schemes. Experimental results for the various filters are presented in the next chapter.

# Chapter 7

# EXPERIMENTAL RESULTS

Experimental results are presented for the test image of Cafe. This grayscale image of 2560 by 2048 pixels contained many edges. A large image size was needed to successfully train and optimize the selection rule $\mathcal{S}$. The simulations consisted of subsampling Cafe using the quincunx lattice without pre-filtering. We interpolated the sub-sampled image back to its original size using the various linear and nonlinear filters defined. We tested the applicability of our Cafe-optimized filter data for $F_{Lin3}, F_{WM2}$, and $F_{EPRS}$, by using it to filter another grayscale image. We selected Aerial, an image of 256 by 256 pixels also with many edges. For these simulation runs, we interpolated a quincunx-subsampled Aerial back to its original size using the data from the Cafe images. For comparison, we also interpolated Aerial using the remaining filters $F_{Lin1}, F_{Lin2}, F_{Med}$, and $F_{WM1}$.

All of the filters defined were applied to the first set of original observation vectors $\mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_8, \mathbf{x}_{12}$, and $\mathbf{x}_{16}$. However, extending the observation vector for linear filters with linear statistics provides no real performance increase. Hence the three linear filters $F_{Lin1}, F_{Lin2}, F_{Lin3}$, as well as $F_{WM2}$, based on the linear optimized weight coefficients, could not be implemented on the extended observation vectors $\mathbf{x}_{4,2}, \mathbf{x}_{8,2}, \mathbf{x}_{12,2}$, and $\mathbf{x}_{16,4}$. The remaining nonlinear filters $F_{Med}, F_{WM1}$, and $F_{EPRS}$ were applied to extended observation vectors as well as the original ones.

## 7.1 Original Observation Vectors

Figure 7.1 through Fig.7.4 presents the results for the original observation vectors. In Fig.7.1, we consider $N = 2, 3, \ldots, 8$. For this simulation run, our initial sample pool size $N$ was small enough to allow us to rank all the samples in the rank pool (we could always implement the upper bound of $M = N$). Figure 7.1-a and Fig.7.1-b give the MAE and the MSE between the original and the reconstructed Cafe. Figure 7.1-c and Fig.7.1-d gives the corresponding error for Aerial, using Cafe's training data for the Weiner coefficients for $F_{Lin3}$ and $F_{WM2}$, as well as the rank optimization table for $F_{EPRS}$. Two EPRS filters are presented for each observation vector; one that was optimized under the MAE criterion (Fig.7.1-a and Fig.7.1-c), and one that was optimized under the MSE criterion (Fig.7.1-b and Fig.7.1-d).

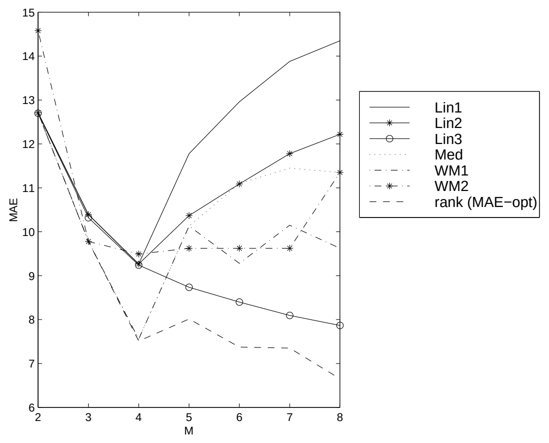
Under both criteria, almost all of the traditional filters' performances peak for $\mathbf{x}_4$. This suggests that the non-optimized linear filters, as well as the median and WM filters all cannot handle the increased samples as the window size increases past 4. In fact, as $N$ is increased beyond 4, only $F_{Lin3}$ and $F_{EPRS}$ perform better. When executed on Cafe, $F_{EPRS}$ performs definitively better than $F_{Lin3}$ under the MAE criterion (Fig. 7.1-a), and nominally outperforms the filter under the MSE criterion (Fig. 7.1-b). When using Cafe-trained data to interpolate Aerial, all the filters peak at $N = 4$ except for $F_{Lin3}$. It seems that $F_{Lin3}$ is outperforming $F_{EPRS}$ as $N$ increases. This is probably due to the over-training of $F_{EPRS}$ on Cafe; as the amount of rank order information in the observation vector is increased, the greater the amount of training needed. However, note when the peak performances of $F_{Lin3}$ at $N = 8$ and $F_{EPRS}$ at $N = 4$ are compared, $F_{Lin3}$ performs only nominally better.

For the observation vectors where $N \geq 8$, it often did not prove practical to rank all the samples in the sample pool for $F_{EPRS}$. Therefore we consider various orders $M = 1, 2, \ldots, N^*$ such that $N^* < N$. Initialy we considered various ordered EPRS filters for $N = 8$ in Fig. 7.2. Note that the upper bound of $M = N = 8$

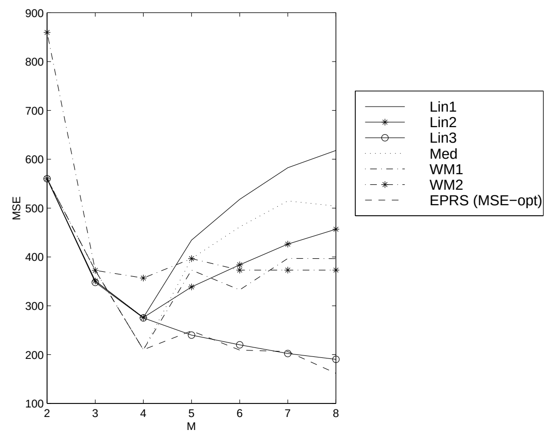
was implemented in the previous simulation run in Fig.7.1. All the other filters' performance are drawn for $N = 8$. When executed on Cafe, $F_{EPRS}$ outperforms all other filters with increased performance as the order is increased from $M = 1, 2, \ldots, 8$. When trained on Cafe and executed on Aerial, $F_{EPRS}$ peaks at order $M = 4$, even at which $F_{Lin3}$ is still outperforming it.

Considering analogous order analysis for $N = 12$ (Fig.7.3) and $N = 16$(Fig.7.4), we see similar results. When executed on the training image, rank interpolation always performs better than all other methods for order 4 and above. When the Cafe training data was applied to Aerial, the rank interpolation outperformed all nonlinear methods for order 4 and above. The only filter to continually outperform the rank filter when applied to a new image was $F_{Lin3}$, the optimized linear filter. An overall performance versus memory analysis for all the $F_{EPRS}$ filters is given in Fig.7.5. We see that the best performance of all our rank filters was for $N = 12$, $M = 6$ when the training image is interpolated. Applying the training data to Aerial, $N = 12$, $M = 4$ was the overall best performer under the MAE criterion, and $N = 4$,$M = 4$ was the best performer under the MSE criterion. However, even in the MAE case, the 12 element observation vector nominally outperformed the much less memory-taxing 4 element observation vector.

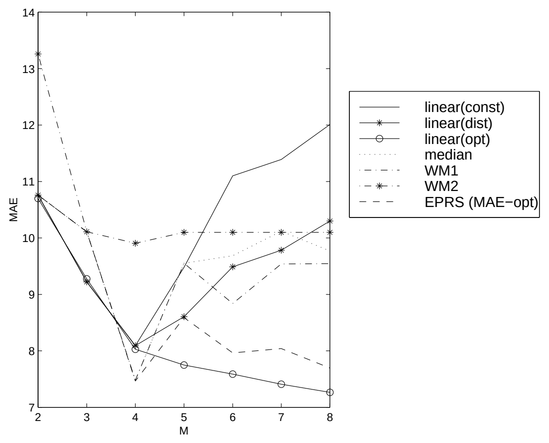
Sample figures are presented in Fig. 7.6. Looking at a subset of Cafe, we consider the difference image between the original and reconstructed images. The results for the various interpolation schemes are presented for $\mathbf{x}_{12}$. Similar results for applying the training data to Aerial are shown in Fig. 7.7. Note when interpolating the training image, $F_{WM2}$ performed better (Fig. 7.6-e), while interpolating Aerial, $F_{WM1}$ performed better (Fig. 7.7-e). We see that the majority of error in all of the various linear and median interpolation schemes stem from the edges within the image, which suggests that the rank interpolation successfully interpolates sharper edges then either method.



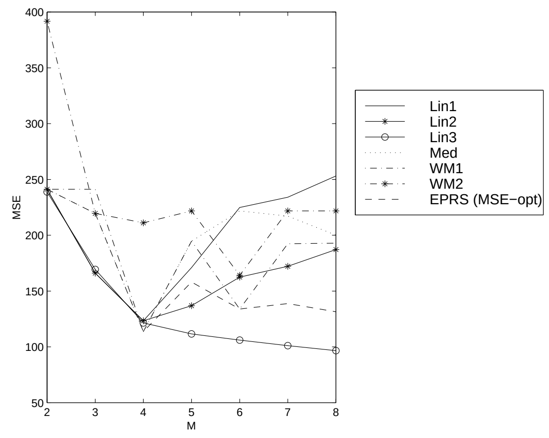
(a)



(b)



(c)



(d)

**Figure 7.1:** Quincunx sub-sampled image interpolation results for original observation vector schemes trained on Cafe. Graphs (a) and (b) show the MAE and MSE results for interpolating the original image, Cafe, using a filter order equivalent to the rank pool size ($M = N$ for $M = 1, 2, \ldots, 8$). Graphs (c) and (d) show the equivalent results for using the training data on the image Aerial.

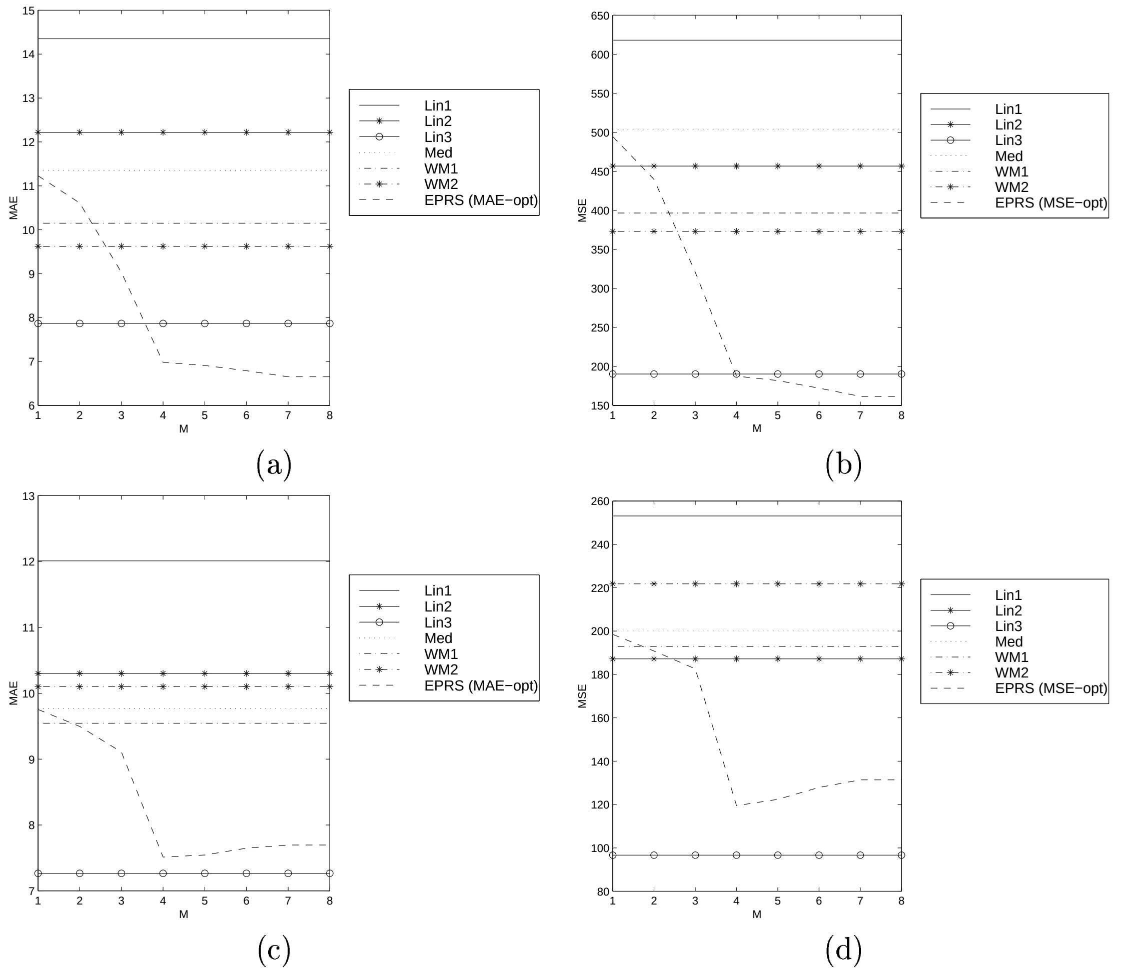
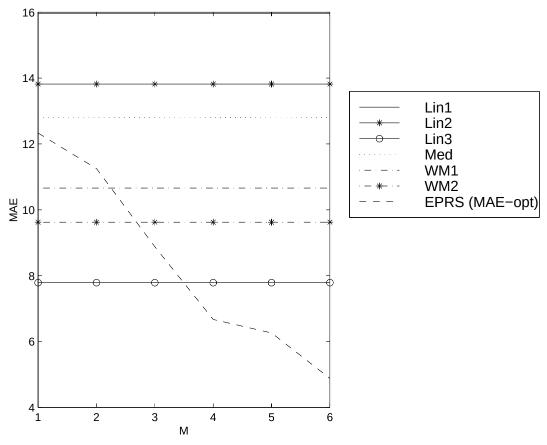
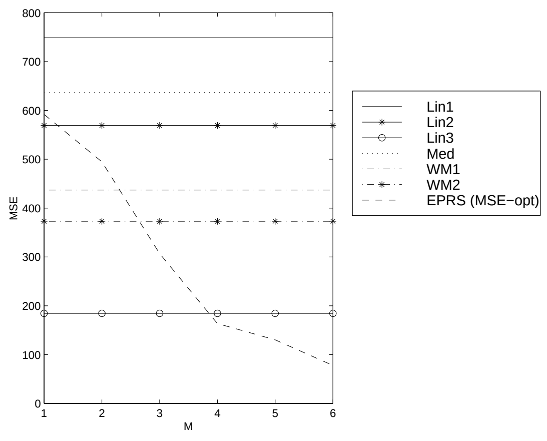


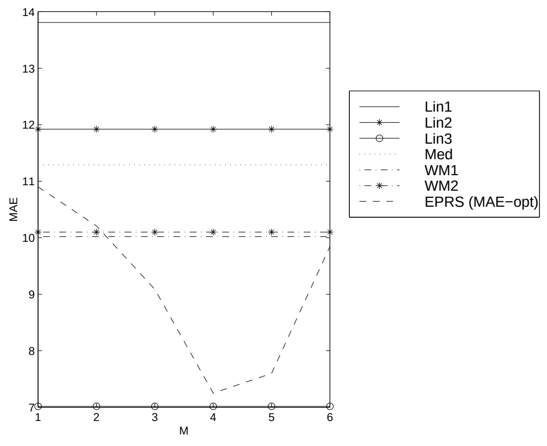
**Figure 7.2:** Quincunx sub-sampled image interpolation results for original observation vector schemes trained on Cafe. Graphs (a) and (b) show the MAE and MSE results for interpolating the original image, Cafe, using a rank pool size $N = 8$. The rank filter order varied from $M = 1, 2, \ldots, 8$. Graphs (c) and (d) show the equivalent results for using the training data on the image Aerial.



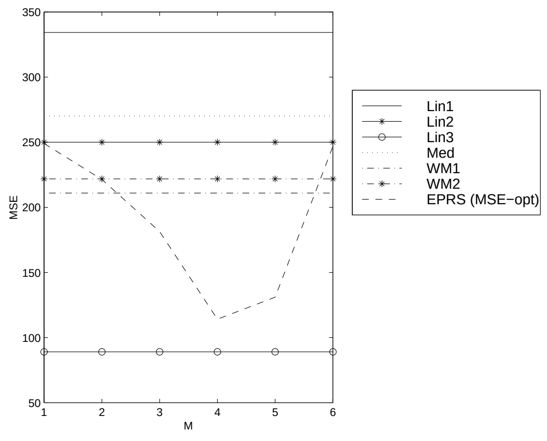
(a)



(b)

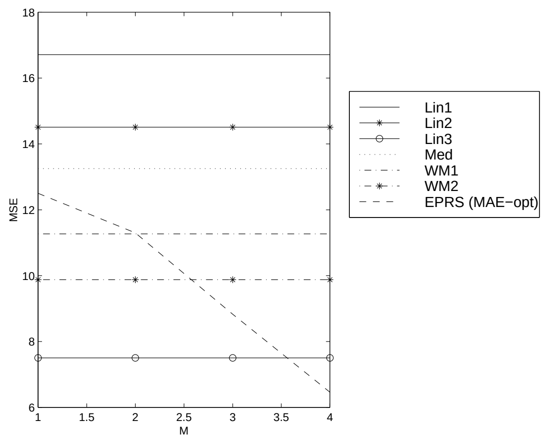


(c)

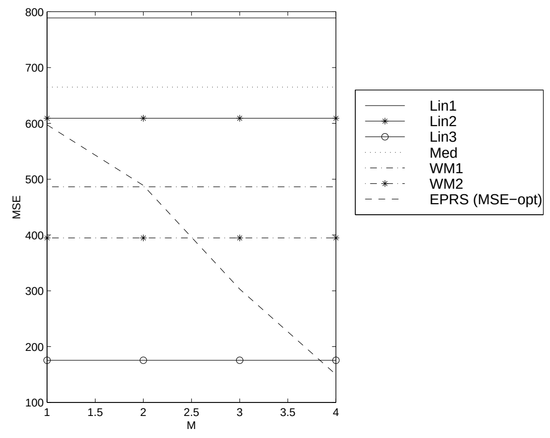


(d)

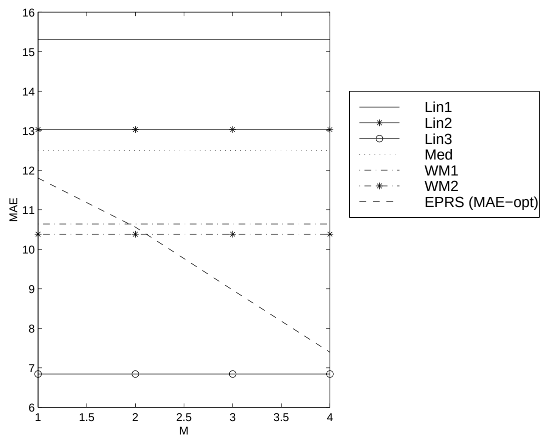
**Figure 7.3:** Quincunx sub-sampled image interpolation results for original observation vector schemes trained on Cafe. Graphs (a) and (b) show the MAE and MSE results for interpolating the original image, Cafe, using a rank pool size $N = 12$. The rank filter order varied from $M = 1, 2, \ldots, 6$. Graphs (c) and (d) show the equivalent results for using the training data on the image Aerial.



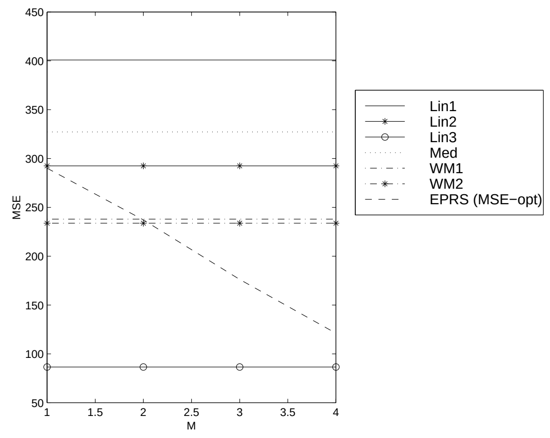
(a)



(b)

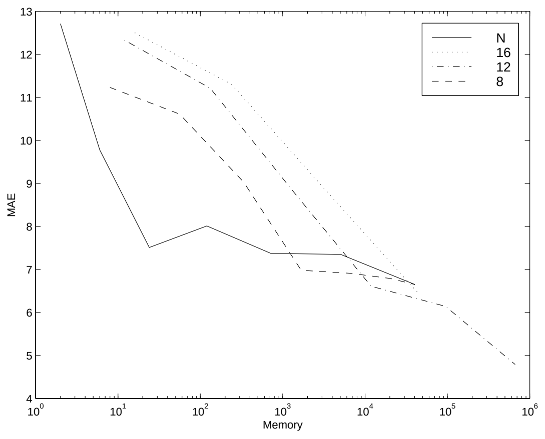


(c)

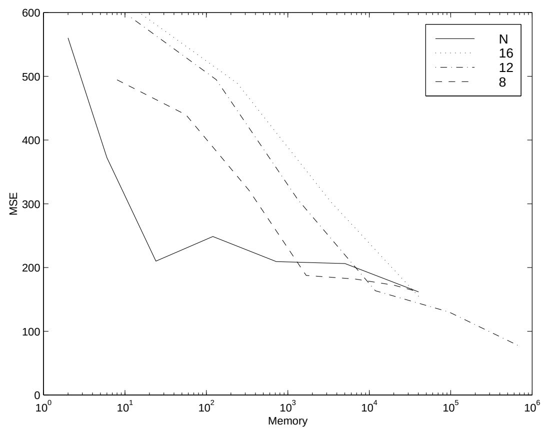


(d)

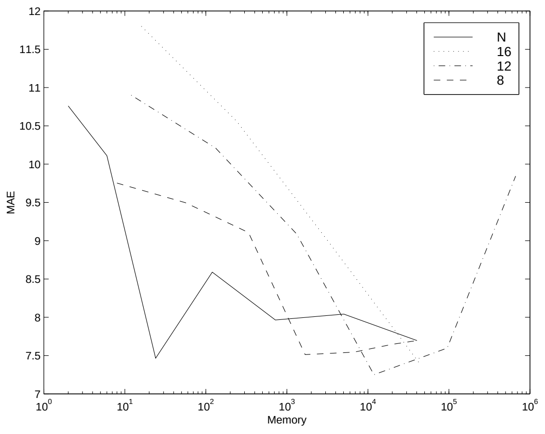
**Figure 7.4:** Quincunx sub-sampled image interpolation results for original observation vector schemes trained on Cafe. Graphs (a) and (b) show the MAE and MSE results for interpolating the original image, Cafe, using a rank pool size $N = 16$. The rank filter order varied from $M = 1, 2, 3, 4$. Graphs (c) and (d) show the equivalent results for using the training data on the image Aerial.



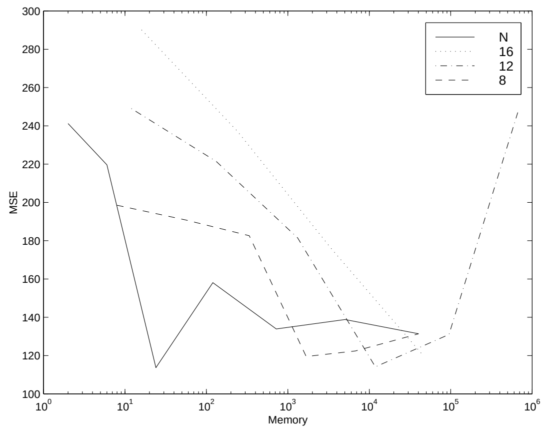
(a)



(b)



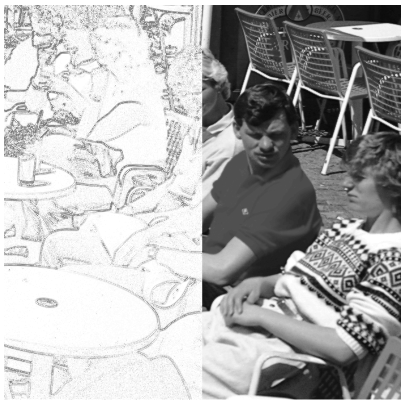
(c)



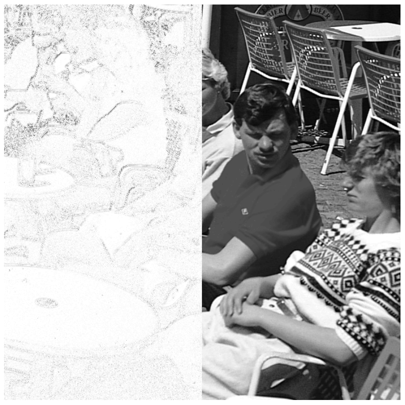
(d)

**Figure 7.5:** Memory-usage analysis (versus performance) of the various rank pools $N$ used for original observation vector rank interpolation schemes. Graphs (a) and (b) show the MAE and MSE results for interpolating the original image, Cafe. Graphs (c) and (d) show the equivalent results for using the training data on the image Aerial.

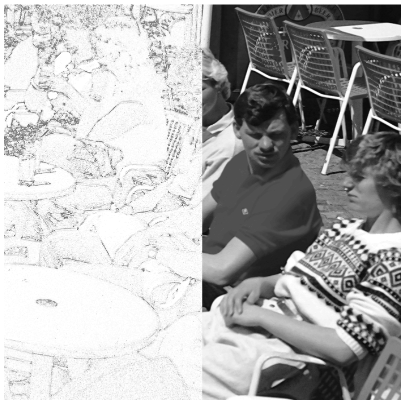
(a)



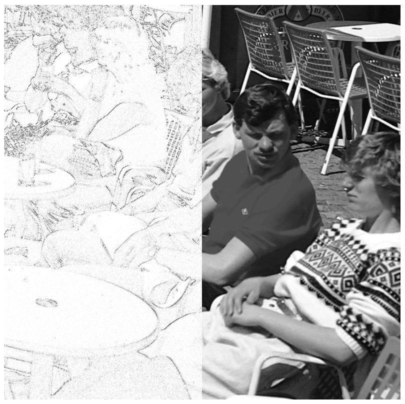
(b)



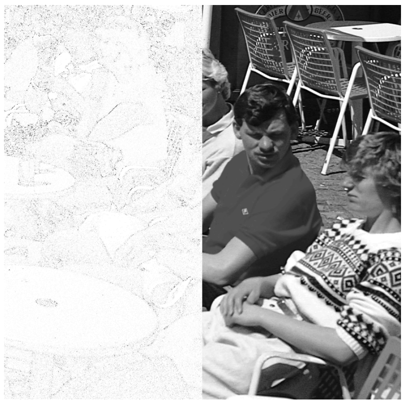
(c)



(d)



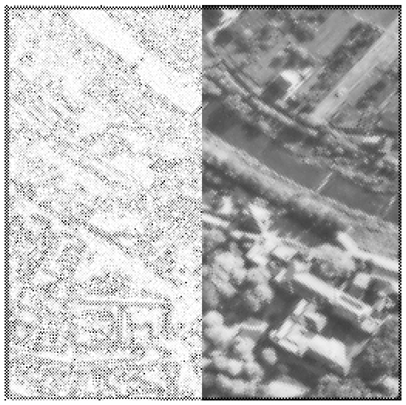
(e)



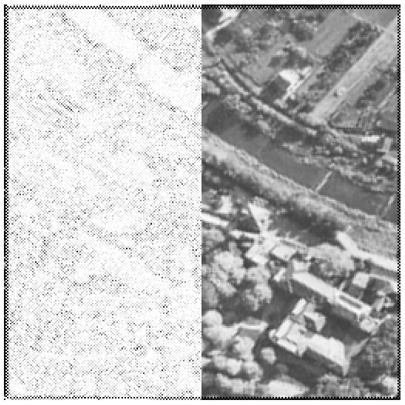
(f)

**Figure 7.6:** Interpolated difference images of quincunx sub-sampled Cafe based on the observation vector $\mathbf{x}_{12}$. (a)Original (b)$F_{Lin2}$ (c)$F_{Lin3}$ (d)$F_{Med}$ (e)$F_{WM2}$ (f)$F_{EPRS}$ with $M = 6$.

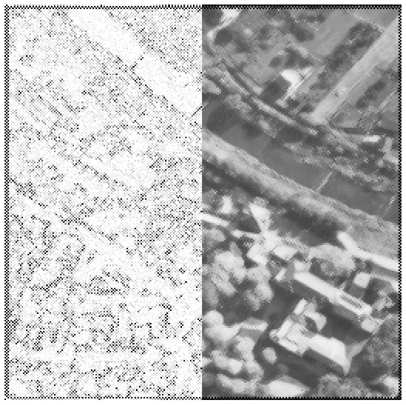
(a)



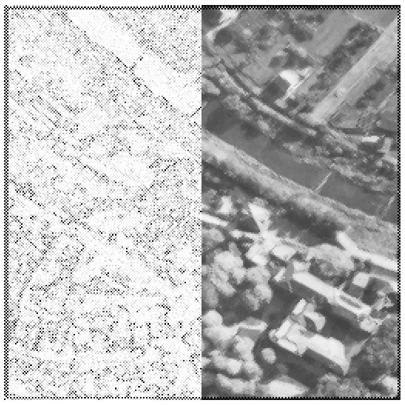
(b)



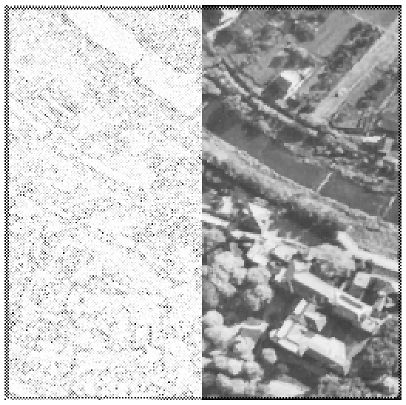
(c)



(d)



(e)



(f)

**Figure 7.7:** Interpolated difference images of quincunx sub-sampled Aerial based on the observation vector $\mathbf{x}_{12}$ using training data from Cafe where applicable. (a)Original. (b)$F_{Lin2}$ (c)$F_{Lin3}$ (d)$F_{Med}$ (e)$F_{WM1}$ (f)$F_{EPRS}$ with $M = 4$.

## 7.2 Extended Observation Vectors

Figure 7.8 through Fig.7.10 as well as Table 7.1 present the results for the original observation vectors. We consider various ordered EPRS filters for $N = 4$ (Fig.7.8), $N = 8$ (Fig.7.9),$N = 12$ (Fig.7.10), and $N = 16$ (Table 7.1). Note that we did not include any linear filters in the extended observation vectors progressions because extending the observation vector by linear statistics provided no performance gains. Analogously, we could not apply $F_{WM2}$ because optimized Weiner filter coefficients could not be derived. For $F_{WM1}$, the weights for the additional $K$ statistics were equal to the mean distance of the averaged samples to the center pixel, as mentioned previously.

For $N = 4$(Fig.7.8), the rank interpolation outperformed all other nonlinear methods for all cases except Aerial interpolation under the MSE criterion (Fig.7.8-d). The seemingly sporadic behavior of this graph is due to the fact that the MSE changed by less than 1% for the various orders. The essentially constant performance seems unsteady due to the magnification of the y-axis.

For $N = 8$(Fig.7.9), the rank interpolation was again the best performer when interpolating the training image. When interpolating Aerial, the rank filters reached a peak at order $M = 4$, where they outperformed the other schemes. Similar results were obtained for $N = 12$ (Fig.7.9), except the peak order for Aerial's interpolation was $M = 3$. This can be attributed again to the under-training of the rank filters at the higher orders. Note that a larger observation window warrants a larger permutation space $\Omega_{\alpha,\beta}$ for the same order as a smaller window, which is the reason why the performance of $N = 12$ peaked at an earlier order than the $N = 8$ progression.

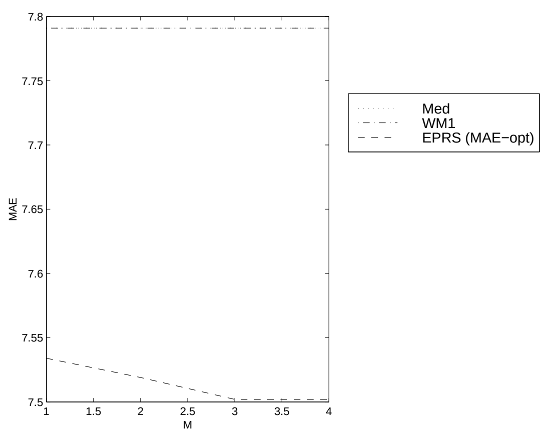
We were only able to execute the $N = 16$ progression up to order $M = 1$ due to memory constraints. The results are presented in tabular form in Table 7.1. When executed on training image Cafe, rank interpolation proves to be the

| Interpolation Filter | Cafe interpolation | | Aerial interpolation | |
|---|---|---|---|---|
| | MAE | MSE | MAE | MSE |
| $F_{Med}$ | 13.15 | 591.8 | 12.22 | 286.8 |
| $F_{WM1}$ | 13.15 | 553.1 | 11.81 | 255.8 |
| $F_{EPRS}$, $M = 1$ | 8.219 | 246.4 | 13.65 | 425.3 |

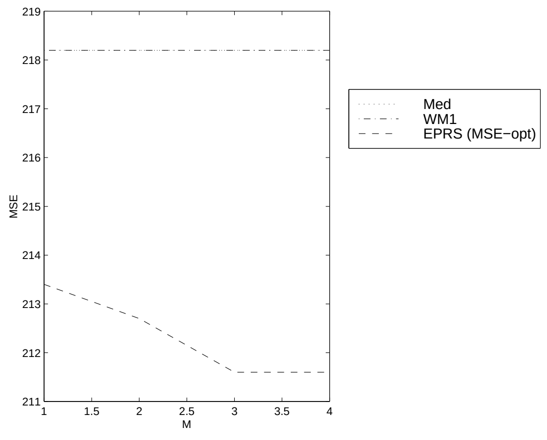
**Table 7.1:** Quincunx sub-sampled image interpolation results for $\mathbf{x}_{16,4}$ vector schemes trained on Cafe. The MAE and MSE results are given for interpolating the original image, Cafe, and the secondary image, Aerial.

optimal nonlinear method, but proves to be the worst when executed on Aerial. This is due again to the large permutation space associated with $\mathbf{x}_{16,4}$, which needed correspondingly more training data to avoid over-optimizing on Cafe.

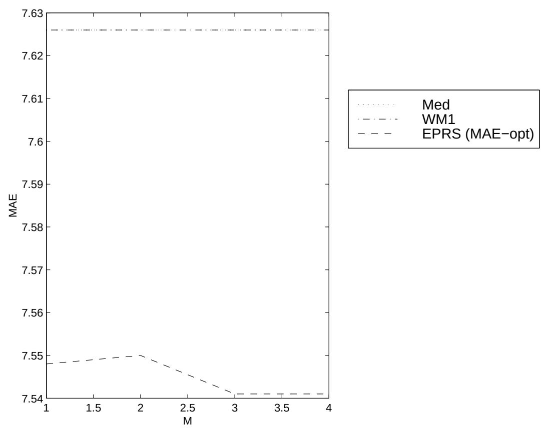
An overall performance versus memory analysis for all the $F_{EPRS}$ filters is given in Fig. 7.11. We see that the best performance of all our rank filters was for $N = 12$, $M = 4$ when the training image is interpolated. Applying the training data to Aerial, $N = 8$, $M = 4$ was the overall best performer under the MAE criterion, and $N = 4$,$M = 4$ was the best performer under the MSE criterion. However, even in the MAE case, the 8 element observation vector nominally outperformed the much less memory-taxing 4 element observation vector. Sample figures are presented in Fig. 7.12. Looking at a subset of Cafe, we consider the difference image between the original and reconstructed images. The results for the various interpolation schemes are presented for $\mathbf{x}_{12}$. Results for applying the training data for $\mathbf{x}_8$ to Aerial are shown in Fig. 7.13. Again, we see that the majority of error in the median-based interpolation schemes stem from the edges within the image, which suggests that the rank interpolation successfully interpolates sharper edges then either method.



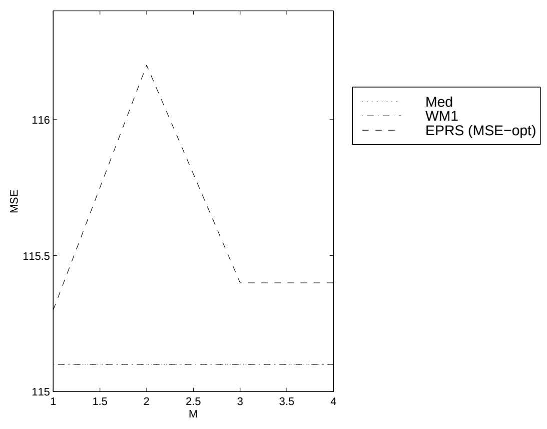
(a)



(b)

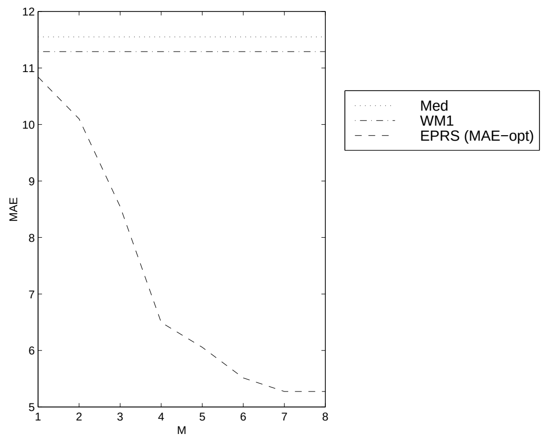


(c)

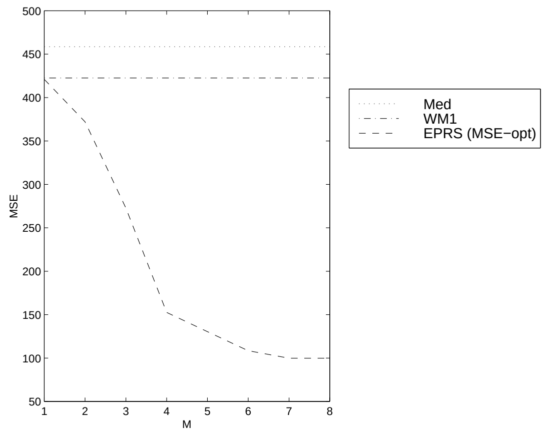


(d)

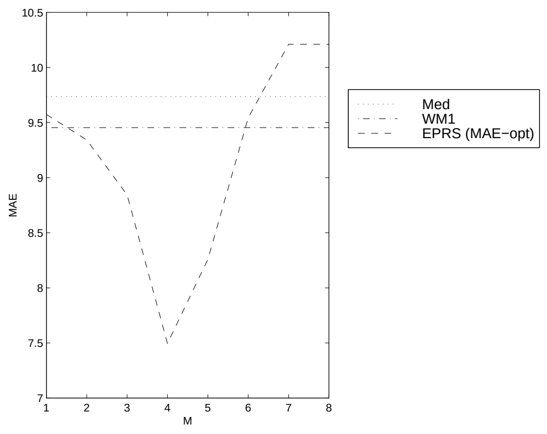
**Figure 7.8:** Quincunx sub-sampled image interpolation results for extended observation vector schemes trained on Cafe. Graphs (a) and (b) show the MAE and MSE results for interpolating the original image, Cafe, using a rank pool size $N = 4$. The rank filter order varied from $M = 1, 2, 3, 4$. Graphs (c) and (d) show the equivalent results for using the training data on the image Aerial.



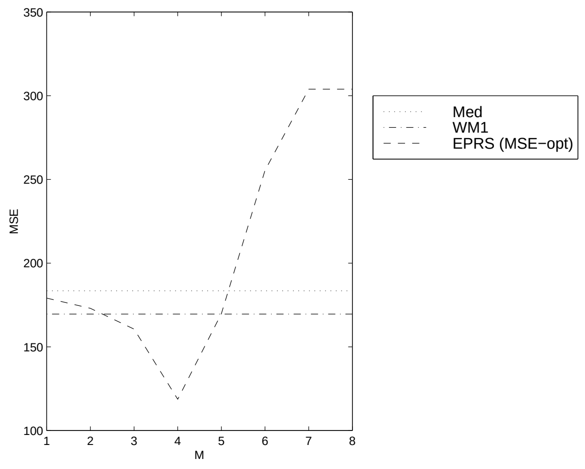
(a)



(b)

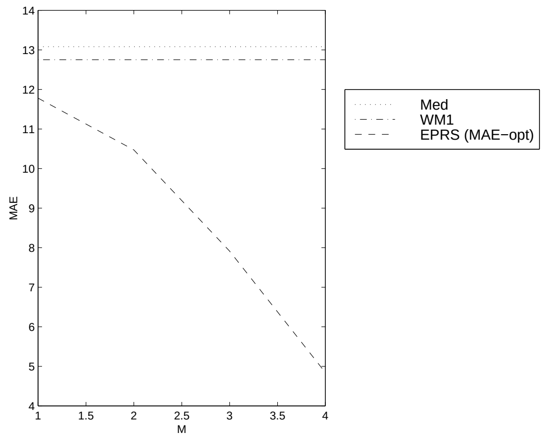


(c)

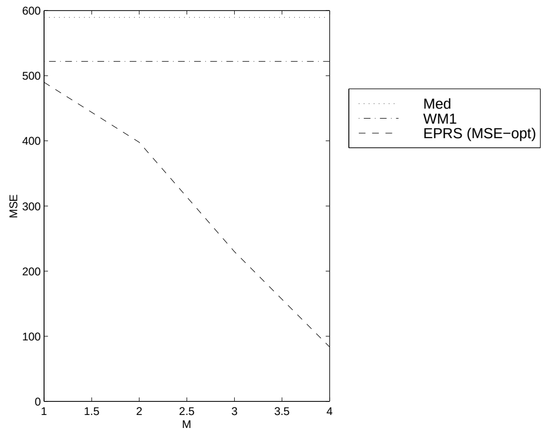


(d)

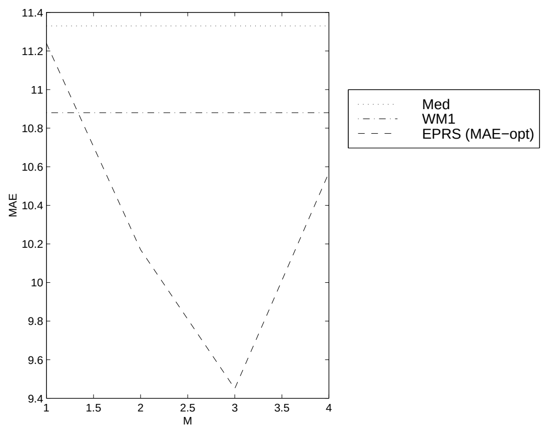
**Figure 7.9:** Quincunx sub-sampled image interpolation results for extended observation vector schemes trained on Cafe. Graphs (a) and (b) show the MAE and MSE results for interpolating the original image, Cafe, using a rank pool size $N = 8$. The rank filter order varied from $M = 1, 2, \ldots, 8$. Graphs (c) and (d) show the equivalent results for using the training data on the image Aerial.



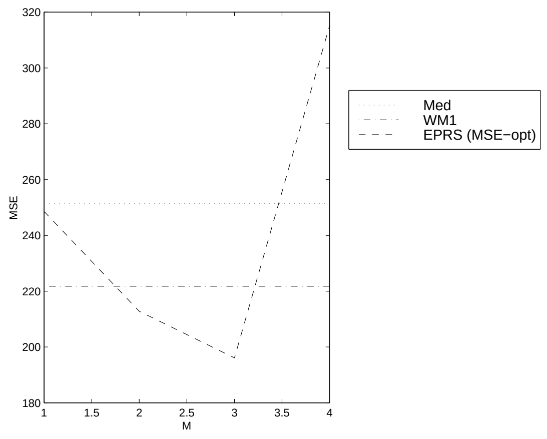
(a)



(b)



(c)



(d)

**Figure 7.10:** Quincunx sub-sampled image interpolation results for extended observation vector schemes trained on Cafe. Graphs (a) and (b) show the MAE and MSE results for interpolating the original image, Cafe, using a rank pool size $N = 12$. The rank filter order varied from $M = 1, 2, \ldots, 4$. Graphs (c) and (d) show the equivalent results for using the training data on the image Aerial.

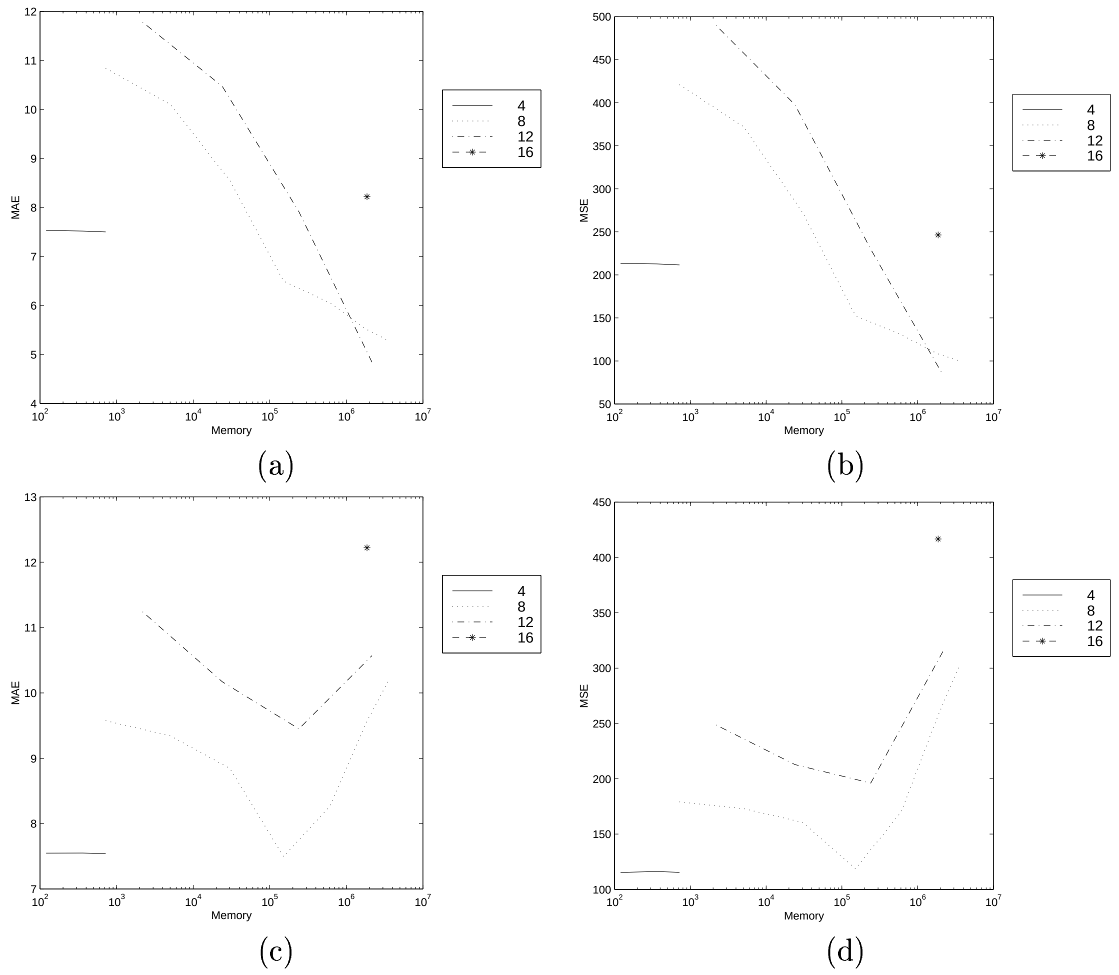


**Figure 7.11:** Memory-usage analysis (versus performance) of the various rank pools $N$ used for extended observation vector rank interpolation schemes. Graphs (a) and (b) show the MAE and MSE results for interpolating the original image, Cafe. Graphs (c) and (d) show the equivalent results for using the training data on the image Aerial.

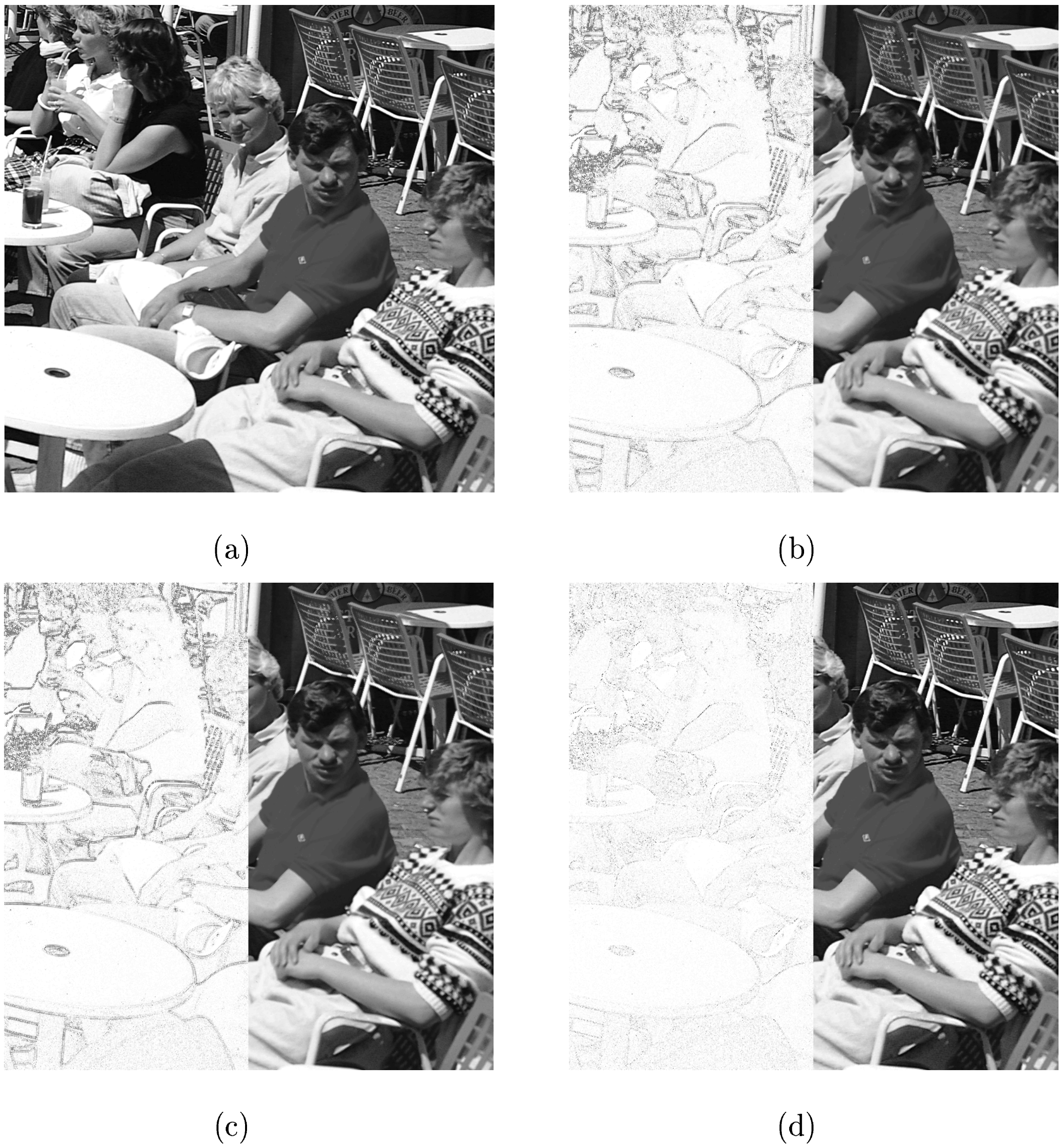
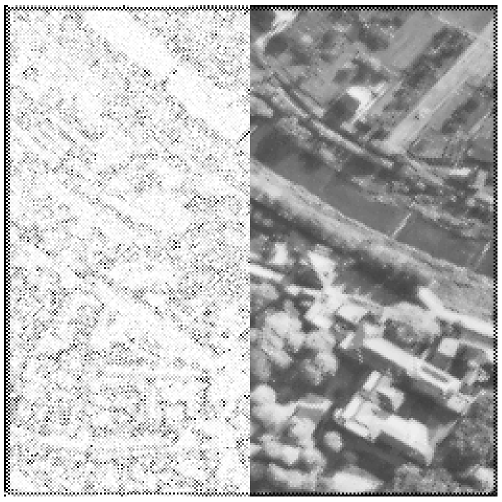
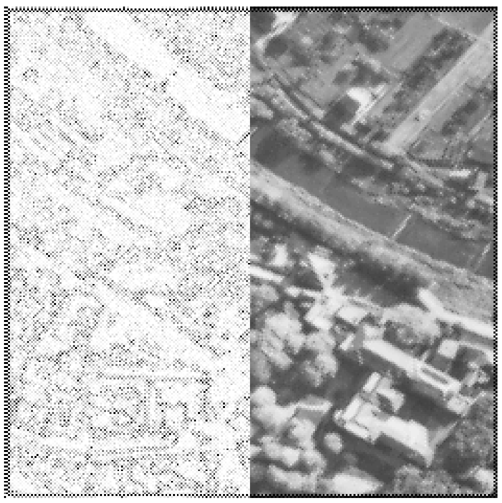


**Figure 7.12:** Interpolated difference images of quincunx sub-sampled Cafe based on the observation vector $\mathbf{x}_{12,2}$. (a)Original (b)$F_{Med}$ (c)$F_{WM1}$ (d)$F_{EPRS}$ with $M = 4$.

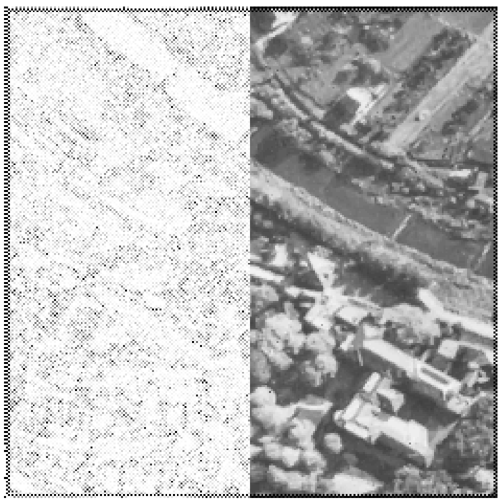
(a)



(b)



(c)



(d)

**Figure 7.13:** Interpolated differenceimages of quincunx sub-sampled Aerial based on the observation vector $\mathbf{x}_{8,2}$ using training data from Cafe where applicable. (a)Original. (b)$F_{Med}$ (c)$F_{WM1}$ (d)$F_{EPRS}$ with $M = 4$.

## 7.3 Conclusions

Generally, we can conclude that the rank ordering information is indeed useful. Furthermore, as the amount of rank-ordered information is increased (the further $\tilde{\mathbf{x}}_{N,K}$ is enlarged by increasing $N$ or extended by increasing $K$), the better the performance of the EPRS filter, assuming sufficient training data. In particular, we note EPRS filters handle the increased sample information better than traditional linear or median methods.

## 7.4 Future Work

The interpolation of the EORC sub-sampling lattice can be examined using the same methodology described in this work. This involves using a polyphase interpolation scheme using three interpolator structures. The use of color images also allows for increased rank order information per pixel [5],[9]. Additionally, in a video transmission scheme, pixels can be rank interpolated along the temporal plane as well as the spatial plane [4]. Based upon our research, rank-based non-linear filters appear promising in all of these areas.

# BIBLIOGRAPHY

[1] G. R. Arce and K. E. Barner. Nonlinear signals and systems. In J. G. Webster, editor, *Encyclopedia of Electrical and Electronics Engineering*. John Wiley and Sons, 1998.

[2] Gonzalo R. Arce, Jose L. Paredes, and John Mullan. Image and video processing handbook.

[3] K. E. Barner and G. R. Arce. Order-statistic filtering and smoothing of time-series: Part ii. In *Handbook of Statistics-16 Order Statistics and Their Applications.*

[4] K. K. Fung and B. Zeng. Spatial/temporal interpolation for very-low-bit-rate video transmission. In *Proceedings of the 1996 (IEEE) Midwest Symposium on Circuits and Systems*, Aug 1996.

[5] R. C. Hardie and G. R. Arce. Ranking in $r^p$ and its use in multivariate image estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 1(2), June 1991.

[6] R. C. Hardie and K. E. Barner. Rank conditioned rank selection filters for signal restoration. *IEEE Transactions on Image Processing*, 3(2), March 1994.

[7] R. C. Hardie and K. E. Barner. Extended permutation filters and their application to edge enhancement. *IEEE Transactions on Image Processing*, 1995. Submitted for publication, December 1995.

[8] S. Haykin. *Adaptive Filter Theory.* Prentice Hall, 3rd edition, 1996.

[9] N. Herodotou and A. N. Venetsanopoulos. Colour image interpolation for high resolution acquisition and display devices. *IEEE Transactions on Consumer Electronics*, 41(4), Nov 1995.

[10] D. L. Kreher and D. R. Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search (Discrete Mathematics and Its Applications)*, chapter 2. CRC Press, 1999.

[11] Chi-Man Lee and Bing Zeng. A novel interpolation scheme for quincunx-subsampled images. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, May 1999.

[12] P. Z. Peebles, Jr. *Probability, Random Variables, and Random Signal Principles.* McGraw Hill, 3rd edition, 1993.

[13] B. Zeng, Y. Neuvo, and A. N. Venetsanopoulos. A class of weighted median filters for image processing. In *Proceedings of the Conference on Information Sciences and Systems*, page 331, 1992.

[14] B. Zeng and A. N. Venetsanopoulos. Compartive study of several nonlinear image interpolation schemes. In *Proceedings of the 1992 SPIE International Society for Optical Engineering*, November 1992.

[15] B. Zeng and A. N. Venetsanopoulos. A split sorting-algorithm for generalized order statistic filters. In *Proceedings of the 1993 (IEEE) International Symposium on Circuits and Systems*, May 1993.