UNIVERSITY OF CALIFORNIA, SAN DIEGO

Pattern Recognition Techniques for Image and Video Post-Processing: Specific Application to Image Interpolation

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical and Computer Engineering

by

Karl S. Ni

Committee in charge:

Professor Truong Q. Nguyen, Chair
Professor Yoav Freund
Professor Gert Lanckriet
Professor William Hodgkiss
Professor Nuno Vasconcelos

2008

The dissertation of Karl S. Ni is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2008

To My Family. I owe you everything.

The true path of success is paved by
diligence inspired by passion.
—Unknown

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The Ph.D. has been an adventure, and I have genuinely depended on several individuals to get through it. Without them, I would not be submitting this thesis today.

I am fortunate to have one of the most understanding advisors, Truong Nguyen, who directed me with general and specific guidance. His efforts in securing funding are much appreciated. I have to thank him for his trust in my abilities because my work is a product of his positive attitude and sincere interest. My goals are set high with his encouragement, and I intend to adopt his management and leadership style throughout my life, professionally and personally. It has been a privilege, truly.

I am also grateful to Professor Nuno Vasconcelos for his part in the development of my thesis. In our conversations, he can best be described as patient and insightful while never short nor curt. I fully appreciate and value his time, which he freely gives even in the presence of his busy schedule. He is always enthusiastic, and I owe a good portion of my publications to his guidance.

Professor Gert Lanckriet is ever the graduate student's friend. His personable manner and his youth provide perspective and make him very approachable. His theories and derivations also contribute to a large portion of my publications.

Someone I often turn to for help is my fellow labmate, Sanjeev Kumar, whose technical abilities are formidable. He aided in several of the algebraic derivations throughout this work. Additionally, I have enjoyed enlightening conversations with Koohyar Minoo, Millie Li, Ryan Prendergast, Ben Kao, and Vikas Ramachandra. I would also like to thank Cheolhong An, Shay Harnoy, Ai-Mei Huang, Natan Jacobson, Yen-Lin Lee, Jack Tzeng, Nick Mueller, Stanley Ho, Gokce Dane, Wade Chang, Mainak Biswas, and Dung Vo. To anyone else that I may have inadvertently missed, my sincerest apologies and my utmost thanks.

My parents, Ming and Ann Ni, are a continuing source of inspiration to me, and I attribute any and all accomplishments to their support. My brother, Kevin, and my sister, Karen, have always been there for me, and they have given me their love, best wishes, and friendship. I can only hope to be as good a brother to them.

Finally, my most profound thanks go to Gina Tuazon, whose love, strength, and encouragement got me through these five years. By building confidence in every capacity of my life, she has given me a strong foundation on which to stand. She is truly an amazing woman.

Portions of Chapter 3 appear in “A Model for Image Patch-Based Algorithms”, Karl Ni and Truong Nguyen, to appear in presentation at the *IEEE International Conference on Image Processing*, October 2008. The dissertation author was the primary author of the publication, and the co-author listed directed and supervised the research which forms the basis for this chapter.

The text of Chapter 4 is adapted from *Adaptive k-Nearest Neighbor*, Karl Ni and Truong Nguyen, January 2008, manuscript submitted to *IEEE Transactions on Image Processing.* Portions of the chapter also appear in “An Adaptive k-Nearest Neighbor Algorithm for Image Interpolation”, Karl Ni and Truong Nguyen, in Proceedings of the *IEEE International Conference on Image Processing*, April 2008. The dissertation author was the primary author of these publications, and the co-author listed directed and supervised the research which forms the basis for this chapter.

The text of Chapter 5 is adapted from *Support Vector Regression based Superresolution*, Karl Ni and Truong Nguyen, in the June 2007 issue of the *IEEE Transactions on Image Processing.* Portions of the chapter also appear in “Single Image Superresolution Based on Support Vector Regression”, Karl Ni, S. Kumar, N. Vasconcelos, and Truong Nguyen, in Proceedings of the *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2006; and also in “Learning the Kernel Matrix for Superresolution”, Karl Ni and Truong Nguyen, in Proceedings of the *IEEE 8th International Workshop on Multimedia Signal Processing*, August 2006. The dissertation author was the primary author of these publications, and the listed co-author directed and supervised the research that forms the basis for this chapter.

The text of Chapter 6 is adapted from *Mixture of Experts Framework for Superresolution*, Karl Ni and Truong Nguyen, January 2008, manuscript submitted to the *IEEE Transactions on Image Processing.* Portions of the chapter also appear in “Kernel Resolution Synthesis for Superresolution”, Karl Ni and Truong Nguyen, in Proceedings of *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 2007; “Complex Function Estimation using a Stochastic Classification/Regression Framework: Specific Applications to Image Superresolution”, in Proceedings of the *SPIE International Conference on Optics and Photonics*, August 2007; and also in “Color Image Superresolution Based on a Stochastic Combinational Classification-Regression Algorithm”, in Proceedings of the *IEEE International Conference on Image Processing*, September 2007. The dissertation author was the primary author of this publication, and the co-author listed directed and supervised the research which forms the basis for this chapter.

The text of Chapter 7 is adapted from *A Zero-phase Filter for Image Interpolation*, Karl Ni and Truong Nguyen, May 2008, manuscript submitted to the *IEEE Transactions on Image Processing.* The dissertation author was the primary author of this publication, and the listed co-author directed and supervised the research that forms the basis for this chapter.

I am very grateful for the financial support of Qualcomm, Inc., the University of California Discovery Grant, and the Von Liebig Foundation. I hope I have represented their contributions well.

VITA

1980	Born, Galveston, Texas
2002	B. S., University of California at Berkeley
2003	Engineer at Lockheed Martin Corporation, Littleton, Colorado
2004	Teaching assistant, Department of Electrical and Computer Engineering, University of California at San Diego
2005	M. S., University of California at San Diego
2008	Ph. D., University of California at San Diego

PUBLICATIONS

A Zero-phase Filter for Image Interpolation. (K. Ni and T. Q. Nguyen) Manuscript to be submitted to the *IEEE Transactions on Image Processing*, 2008.

Mixture of Experts Frameworks for Image Superresolution. (K. Ni and T. Q. Nguyen) Manuscript submitted to the *IEEE Transactions on Image Processing*, 2008.

An Adaptive k-Nearest Neighbor Algorithm for MMSE Image Interpolation. (K. Ni and T. Q. Nguyen) Manuscript submitted to the *IEEE Transactions on Image Processing*, 2007.

Image Superresolution Based on Support Vector Regression. (K. Ni and T. Q. Nguyen) *IEEE Transactions on Image Processing*, Vol. 16, No. 6, June 2007.

"A Model for Image Patch-Based Algorithms". (K. Ni and T. Q. Nguyen), to appear in presentation at the *IEEE International Conference on Image Processing*, June 2008.

"An Adaptive k-Nearest Neighbor Algorithm for Image Interpolation". (K. Ni and T. Q. Nguyen) In the Proceedings of the *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 2008.

"Color Image Superresolution Based on a Stochastic Combinational Classification-Regression Algorithm". (K. Ni and T. Q. Nguyen) In the Proceedings of the *IEEE International Conference on Image Processing*, September 2007.

“Complex Function Estimation using a Stochastic Classification/Regression Framework: Specific Applications to Image Superresolution”. (K. Ni and T. Q. Nguyen) In the Proceedings of the *SPIE International Conference on Optics and Photonics*, August 2007.

“Filling Time by Plugging Holes”. (V. Ramachandra, K. Ni, T. Q. Nguyen) In the Proceedings of the *International Conference and Exhibition on Computer Graphics and Interactive Techniques*, August 2007.

“Kernel Resolution Synthesis for Superresolution”. (K. Ni and T. Q. Nguyen) In the Proceedings of the *IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2007.

“Learning the Kernel Matrix for Superresolution”. (K. Ni and T. Q. Nguyen) In the Proceedings of the *IEEE* 8^{th} *International Workshop on Multimedia Signal Processing*, August 2006.

“Single Image Superresolution Based on Support Vector Regression”. (K. Ni, S. Kumar, N. Vasconcelos, and T. Q. Nguyen) In the Proceedings of the *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2006.

ABSTRACT OF THE DISSERTATION

Pattern Recognition Techniques for Image and Video Post-Processing: Specific Application to Image Interpolation

by

Karl S. Ni
Doctor of Philosophy in Electrical and Computer Engineering

University of California San Diego, 2008

Professor Truong Q. Nguyen, Chair

Video and image enhancement relate to the age old problem of constructing new image detail to effect a more pleasurable viewing experience in video sequences and images. The image processing procedure described in this thesis is image and video interpolation, but the discussed algorithms are easily applicable to other areas in a variety of fields. The motivations behind resolution enhancement span the full spectrum of government to commercial to medical applications. For example, video quality from on-demand sites such as YouTube, Daily Motion, and Veoh, where hit counts sometimes reach 4.7 million viewers per day, have traded off quality for bandwidth; travelers who use Google Map's Street View cannot read signs, see landmarks from the provided low-resolution panoramic; Google Earth® and government contractors in image intelligence demand for increasing resolution in satellite imagery. The long list of applications goes on, including applications such as web-photo editing, MRI scan feature enhancements, on-demand creation of HDTV content, etc.

The image interpolation problem is an inherently ill-posed problem, and quality assessments of the resulting images are often driven by human decision rather than numerical analysis. By computer learning, the best viewing performance can be achieved by mimicking the human decision-making process. Therefore, this thesis is concerned with modeling machine and statistical learning techniques to fit the interpolation framework, while easily modified to accommodate other video and image processing problems in general. Specifically, we learn properties in a training set by using regression to map a relationship between known and unknown information. The known information by our definitions are low-resolution images while unknown information are high-resolution images.

Machine learning is a particularly broad topic, and we can approach the image interpolation problems in several different ways. Our input space is the image patch domain, where we process fixed-size contiguous subsets of the image independently. Consequently, we first discuss properties of the feature space and propose a multivariate probability distribution function to describe the image patch domain. Knowledge of the distribution and properties of the feature space is especially conducive to both parametric and nonparametric estimation techniques.

Because k-nearest neighbors is a relatively simple and commonly used non-parametric estimation technique, we propose a nearest neighbor algorithm that adaptively finds the appropriate number of neighbors to use and then performs the necessary regression steps. The algorithm also imposes global constraints through a heavily approximated Markov Random Field. Due to runtime considerations, we explore quicker ways to search for the k nearest neighbors of a given input.

Next, we investigate the use of kernel-based methods by employing support vector regression. We improve the generalization potential for nonlinear relationships by proposing convex optimization problems focused in kernel learning. There are various interpolation frameworks that can include the newly proposed regression technique, and we experiment with several. Ultimately, we propose a mixture of experts framework to describe the relationships in the training set.

Finally, we propose a single, general, zero-phase MMSE interpolation filter to address computational complexity concerns of all learning algorithms. The idea arises from image processing analysis of machine learning techniques rather than the application of machine learning to image processing. In the development of the final filter, we analyze a general classification-based filtering scheme using polyphase representation. Because there are inherent similarities and considerable overlap between each class in such an approach, one zero-phase filter for all image

content seems to logically follow as an adequate approximation that reduces the total number of computations to that of bicubic interpolation. Analyzing the frequency response, we can generate filters on-the-fly for arbitrary scaling factors.

1 Introduction

It is not knowledge, but the act of learning, not possession but the act of getting there, which grants the greatest enjoyment.

—Carl Friedrich Gauss

Human beings have survived for millions of years in large part by our ability to recognize patterns and draw inferences from them. In an attempt to comprehend what is involved in generalizing to real-world phenomena, computer scientists study statistical and machine learning principles. The underlying methodology behind statistically modeling data is to extract functional relationships from a controlled database and apply these relationships to unseen test points. In essence, based on what has been previously observed, we apply that information to the problem at hand to make an informed decision. The block diagram can be seen in Fig. 1.1.

To recognize patterns, humans use their primary senses to gather information about the problem at hand. One collection of inputs can be extracted from the eyes and vision, where decisions are often based off of visual cues. How we view the world from the external scene outside the eye to the brain's interpretation of a scene has been grouped into a broad and complex model called the human visual system, which has been extensively studied in the biological [102] and analytical sense [22, 110, 76]. Two fields of studies are dedicated to the automation of decision making processes through the human visual system: computer vision and image processing.

There is not a fine line distinguishing computer vision and image processing, and collaboration often occurs to cover the overlap between the two fields. Computer vision deals with problems like object recognition and three-dimensional

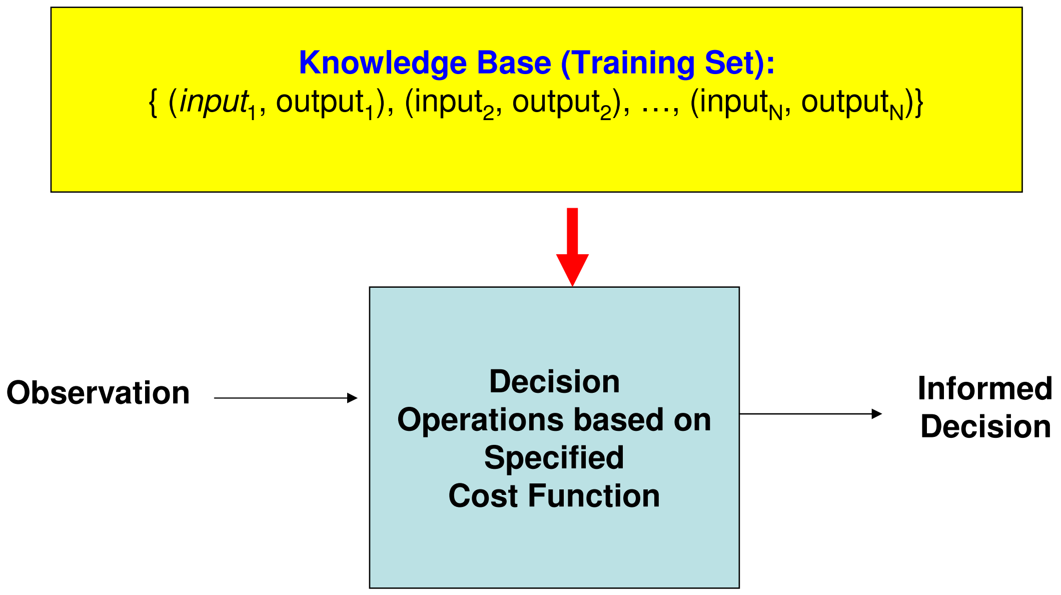

Knowledge Base (Training Set):
{ (input$_1$, output$_1$), (input$_2$, output$_2$), …, (input$_N$, output$_N$)}
Observation
Decision Operations based on Specified Cost Function
Informed Decision

Figure 1.1: Statistical and Machine Learning General Framework

information extraction from a two-dimensional signal. Problems in computer vision usually stem from the higher-level processes within an image. In contrast, image processing is concerned with the two-dimensional properties of images, with emphasis on the image itself. In such a context, lower-level terms like “edges”, “texture”, and “image segmentation” are relevant and tangible concepts.

Whereas computer vision fully embraces pattern recognition and statistical and machine learning, the equivalent effort in image processing in the past has largely been left unexplored, and only recently have conferences been dedicated to the topic (see EURASIP’s Special Issue on Machine Learning in Image Processing). Despite the limited scientific exposure, it seems only natural to utilize learning because we, as humans, use learning to assess or process images. Our evaluation of images is highly subjective because numerical metrics cannot adequately assess an image on a global scale. Thus, when the underlying solutions mimic how human learning would process the image, the results should reflect a more favorable HVS evaluation, or in essence, effect a more visually pleasing image. For example, given the blurry block on the left in Fig. 1.2, our intuition should be able to create the image on the right by applying what we know of edges and lines from previous experiences. Similarly, given a training set, we hope to automate the decision-making process. By utilizing a training set of a priori knowledge, the image on the right can be created.

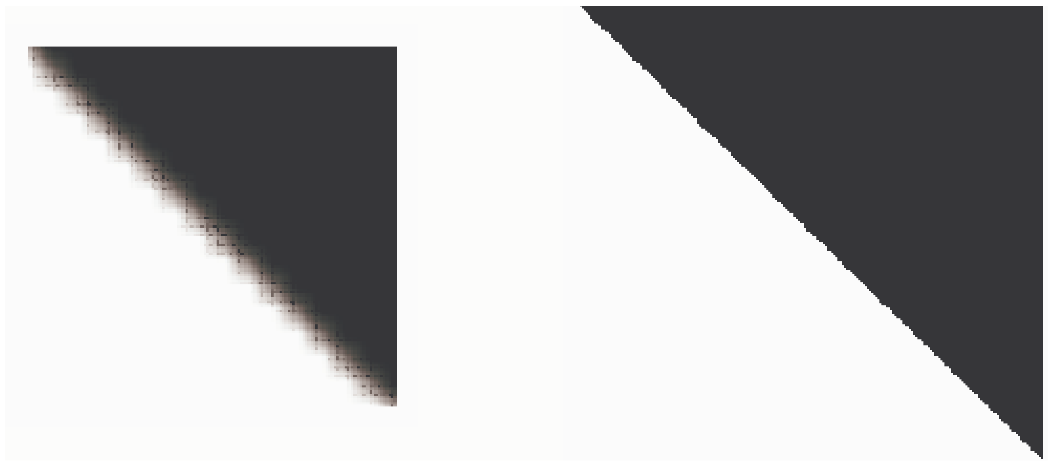


Figure 1.2: The Power of Human Inferences

This thesis sets out to establish good learning techniques in relation to the specific problem of image interpolation and superresolution, one of the more basic image processing problems. The functional relationship between low and high-resolution images is estimated through regression, and we explore various frameworks to do so. Although regression solves our particular problem well, we have deliberately chosen a general format for applicability to other imaging problems. We view our designs of function estimation as a tool rather than a means to a sole end.

1.1 Towards a Shift in Computational Focus

Image interpolation, sharpening, deblurring, and other issues in video post-processing collectively have long since been a problem whose goal is to provide a viewer with "acceptable" high-resolution image results (be it visually pleasing, numerically correct, or some other criterion). Yet, only recently have developments in digitization, displays, capture devices, information technology, and benefits reaped by Moore's law provided significant demand and renewed interest for resolution enhancing solutions. For example, Reuter's reported that in a press release from comScore (NASDAQ: SCOR), one of the leaders in metrics of the digital world, the United States alone watched more than ten *billion* online videos during the month of December 2007, the heaviest consumption to date. The number corresponds to an astounding 76.7% of all U.S. internet users not just visiting websites, but sitting down and watching videos for extended periods of time. The proliferation of videos and images in various technological settings can be seen in Fig. 1.3, where inevitable needs for video post-processing arise.

Figure 1.3: Recent Technological Needs for Image Processing

The most recent high-profile acquisition that involves content that is directly related to video and image processing is Google's buyout of YouTube. Seeing the potential for advertisements in the video arena, Google sites have extended their market share to one third (32.6%) of all videos viewed online. The most popular video postings boast hit counts of up to 4.7 *million* viewers *per day*. Internet websites such as Dailymotion, Veoh, Metacafe, Tudou, etc., have followed suit offering videos of different lengths and categories. Such sites have traded off video quality for bandwidth, compressing videos excessively in favor of quick download times so that the viewer receives a more on-demand experience. The remedy for highly compressed, poor video quality is video post-processing at the receiving terminal, where we build filters for video superresolution, deblocking, denoising, and deblurring.

Cameraphones are also shown in Fig. 1.3, where integration of all things portable and electronic, including digital cameras, have merged into a single device. The obvious improvement is increasing resolution on cameraphones, but there are other future video processing projects that will inevitably surface. Although not yet popular, streaming video capability in cellphones has already begun to make its mark with brands like Verizon's VCAST ®and Sprint's Power Vision ®. Additionally, cameras on phones have opened the door to video teleconferencing. Soldiers will talk to their newborn children overseas, presentations will be regularly given remotely, and families will be visually connected across the globe; their demand for high-quality video will require video postprocessing to make it all happen.

Fig. 1.3 emphasizes image and signal intelligence, where military has heavily relied on satellite imagery. Funding for specific projects include the design of a group of filters assembled to highlight a target area of interest. In the commercial sector, Google Maps, also sporting a satellite view option, has found it necessary to introduce "Street View" to supplement existing top-down views, because users sometimes rely on visual landmarks. The resolution, however, is too poor to read street signs or house numbers, which users may need to identify areas of interest. Image interpolation, the main application of our machine learning algorithms, resolves this problem very well.

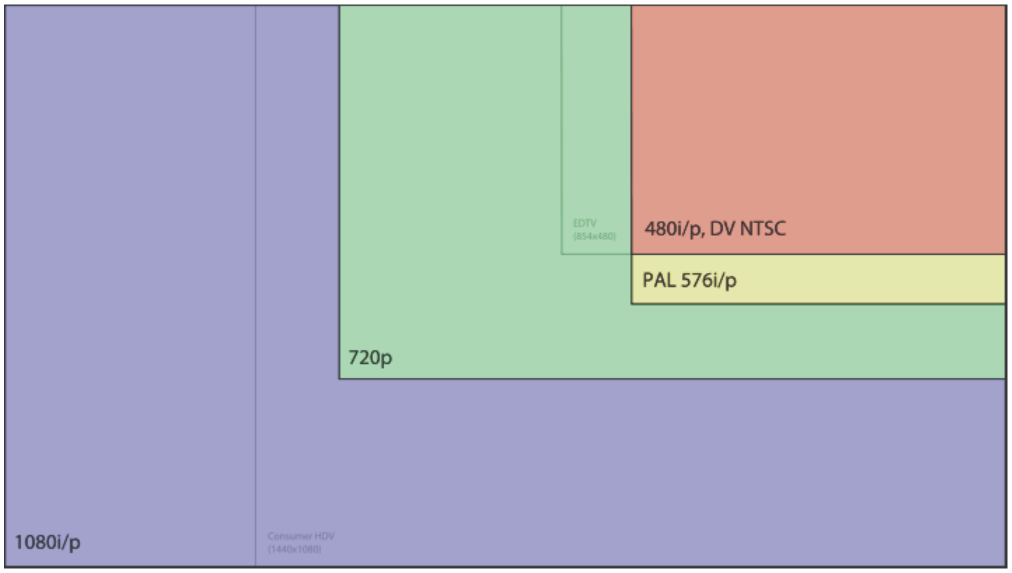


Figure 1.4: Typical Video Formats

Finally, the trend in large screen displays states that support for super-high definition now exists. Common video formats shown in Fig. 1.4 are typically broadcasted at a maximum of 1080p (or $1K$), but display technology can now support $2K$ and $4K$ pixels. Naturally, few video sequences exist at $2K$ and $4K$, so any movie on the large displays will appear blurry or undersampled. To bring lower-resolution movies to these large displays, we require a quick algorithm to superresolve to high resolution.

In less mainstream applications, specialized fields with needs pertaining to image and video processing include those of biomedical imaging. While collectively, doctors and patients are not yet ready (nor do they figure to be in the near future) to automate diagnosis and detection, there has been considerable legal paperwork in this respect, where the United States Food and Drug Administration has already approved the use of the new computer aided detection device "MammoReader" software. Although detection in such cases has often been exclusively associated with the computer vision realm, there is a legitimate argument that finding the optimal features, where image attributes can be highlighted either by filter or other image transformation methods, is more important. The filters, potentially obtained in a manner similar to those described in later sections, are designed to elucidate the defining characteristics of an object.

1.2 Image Interpolation

Image interpolation relates to methods of constructing new image detail from a discrete set of known points resulting in a higher resolution image. The problem is ill-posed, and the quality of the solution is usually considered subjectively, focusing on edges, texture, and clarity of content. These properties can be generated in a number of ways, but to obtain them, new or assumed information must be introduced. The information can come in many forms, including but not limited to assumptions on pixel properties [74, 62, 3], frequency properties [4, 2], a set of low-resolution, shifted images [83, 123, 16], or a training set in statistical and machine learning [9, 49, 28, 38, 24, 85, 84, 86, 87, 88]. In our work, it is the final category in the list, and the proposed algorithm is concerned with its application in a sliding-window approach.

1.3 Focus of This Work

Processing the entire image at one time in a single non-iterative pass is ineffective, computationally expensive, and unmanageable in terms of dimensionality. Hence, the proposed algorithms are patch-based. The domain of image patches has specific and interesting properties, and we explore its potential as an input space by building a distribution model in Chapter 3. The model hinges on a combination of a multivariate super-Gaussian distribution and the commonly used Gaussian Mixture Model, and some intuition about what the mixtures actually mean is explained.

Our contributions are a group of interrelated image interpolation algorithms. There are four main algorithms, and they are listed as

- Adaptive k-Nearest Neighbor Tied with a Markov Random Field

- Support Vector Regression in the DCT Domain
- Mixture of Modified Support Vector Regressors
- A Single, Zero-Phase Filter Built from Polyphase Representation

The list of proposed algorithms estimate functional relationships between input and output space. Although the specific goal is image superresolution, the learning technique is intentionally broad in application to span relationships that may be learned for image denoising, deblurring, and demosaicking.

To arrive at the final form for each of the interpolation techniques, we set up some optimization problems, most of them convex. Optimizations include (1) finding the correct value of neighbors to use in k-Nearest Neighbor, (2) learning a kernel matrix for support vector regression after reading the thesis, and (3) adapting the support vector regression to weight "relevant" data. There are many derivations throughout the body of the work, but the aforementioned optimization techniques are easily generalizable to arbitrary problems. The chart of items to be discussed and their categorization is given in Fig. 1.5.

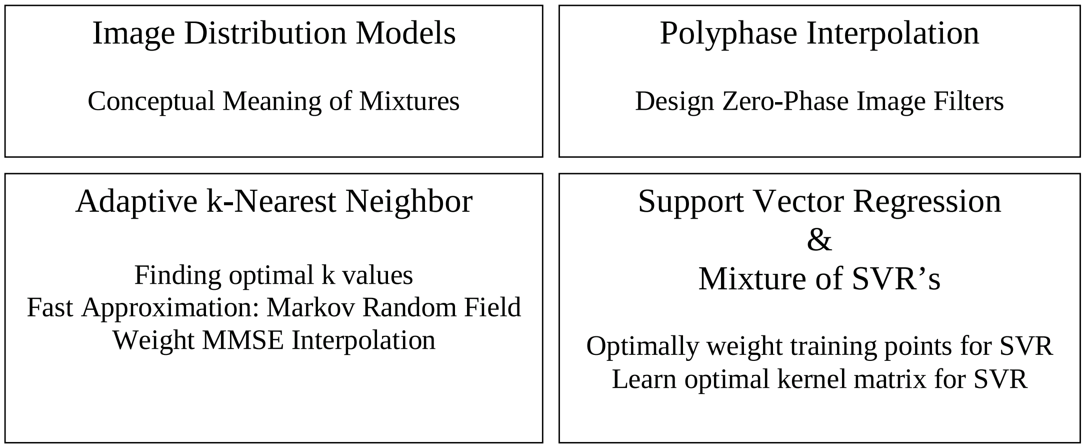


Image Distribution Models
Conceptual Meaning of Mixtures
Polyphase Interpolation
Design Zero-Phase Image Filters
Adaptive k-Nearest Neighbor
Finding optimal k values
Fast Approximation: Markov Random Field
Weight MMSE Interpolation
Support Vector Regression
&
Mixture of SVR's
Optimally weight training points for SVR
Learn optimal kernel matrix for SVR

Figure 1.5: Topics to be Discussed in Subsequent Chapters

2 Background

A study of the history of opinion is a necessary
preliminary to the emancipation of the mind.

—John Maynard Keynes

Image interpolation and superresolution is one of the oldest problems in digital image processing. It is also the simplest problem to understand and perhaps the most difficult to solve. Because of its broad applicability to images on the whole, the collection of solutions to the image interpolation problem has accumulated a diverse and rich history. In this chapter, we explore this history and related work in the video processing field specifically related to image interpolation.

To capture images, a scene is sampled and stored as a discrete collection of points. The need to interpolate arises when when we would like to increase the number of samples in the original set of points. The solution is to fill in samples on a more concentrated, regular grid. The original image is described as the low-resolution image. The end-image is called the high-resolution image. The procedure by which the high-resolution image is created from the low-resolution image is the definition of our problem.

It is well known from image restoration theory that image interpolation is an ill-posed problem. The most common solutions [57, 119, 81, 66] are splines methods, where polynomial functions and their derivatives are used to approximate 2-D function behavior with surrounding pixel values. As will be seen in Chapter 4, splines interpolation methods perform adequately, but their results are mediocre at best due to their inability to adapt to image content. Using splines is the "safest" solution because of their tendency to average the data out. Consequently, no additional frequencies are being added, and the image data is smeared over a larger spatial grid. The result is a blurry image whose pixel values are not estimated, but rather represent the contribution of surrounding, known pixel values.

Splines have long been considered and used as a standard for comparisons in quality and complexity. Their use is not limited to image interpolation, as they have been used for function estimation in audio and one-dimensional signals first. As the demand for higher resolution images grew stronger, several image superresolution algorithms surfaced. The purpose of this chapter is to discuss those interpolation methods that are relevant and perform especially well for comparison purposes in later chapters. Non-statistical learning methods are reviewed in in Sec. 2.1, and then statistical learning methods similar to ours are described in detail in Sec. 2.2.

2.1 Non-statistical Superresolution Methods

Aside from typical spline methods, several approaches interpolate in other domains, including the DCT domain [4, 2], DWT domain [117, 120], and Fourier Domain [63, 101]. Alternative domain methods usually perform some type of zero-padding in higher frequency slots, which after taking the inverse transform, results in a spatially larger image. However, instead of superresolution the result exhibits characteristics more of re-*scaling*, where higher resolution information is not added, i.e. edges and texture are not elucidated, but again, existing information is spread out over a larger spatial area.

More complex classes of solutions use range-based operations (content-based interpolation), methods that elucidate edges and borders while maintaining texture continuity. The simplest among such solutions is bilateral filtering [116, 129]. Extensions to [116] include edge directed interpolation techniques [62, 5, 74]. [74] uses a low-resolution correlation matrix as an approximation to obtain a high-resolution image filter based on an assumption of geometric duality. Although simple to implement, the covariance matrix is still low-resolution, and the value added is usually inadequate for complicated textures, often causing an effect similar to aliasing. The methods are basic in nature and do not enhance resolution especially well, but their existence justifies some serious thought when determining an optimal feature space, an idea central to many learning-based algorithms.

Algorithms that concentrate on particular image attributes often preserve some type of regularity [115, 26, 25] including measures that are tailored specifically to edges. Modeling items within images as polygons and geometric figures [49] preserves edges and shapes, but most image content is usually too complex for simply assuming that shapes within an image are only polygonal. [26] uses properties in the decay of wavelet coefficients to predict unknown coefficients at higher resolution subbands. Again, the assumptions made in [26] may be insufficient when considering all types of image content. In fact, in terms of generating resolution, such algorithms are inferior to most statistical learning techniques where the quantity of additional available information (provided a priori through a training set) may be exceedingly large. Therefore, to compare the two types of algorithms may be unfair because the information available to non-learning-based algorithms is considerably less.

In considering more sophisticated methods where information is more abundant, inputs can be defined as shifted and warped versions of each other with computed shift and warp parameters. Bayesian reconstruction techniques such as MAP, ML, IBP, POCS [83], [108], [17], [123], [98] study the reverse problem to obtain the high-resolution image that could have produced a group of observed, similar test images. The approach in our work considers a more limited input space, using inferences instead of existing inputs.

2.2 Learning-Based Superresolution Methods

Learning-based algorithms find relationships by drawing inferences from a training set and generalizing to unseen data points. The inferences can be made from a set of low and high resolution image patch pairs. Relationships can then be generalized to unknown data points or whatever can possibly be used as input.

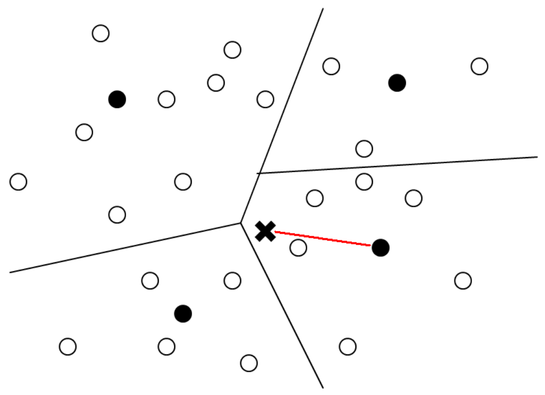
Most frequently cited methods for superresolution include example-based superresolution in a Markov network [49], manifold learning through neighbor embedding [28], and various other statistical learning approaches [79, 38]. Other statistical learning algorithms involve outdated neural network concepts [38] with limited degrees of success in interpolation in terms of visual quality.

In our experimentation, the most successful algorithm[1] is the classification-based adaptive filtering in [9]. The problem is formulated by using localized choices for filters separated by the EM algorithm [39]. Under the assumption that images are distributed as a Gaussian mixture, [9] applies content-based interpolation by applying class labels to each mixture. The domain is partitioned according to image content, and problem complexity decreases per test point, which yields good performance. Because the domain is partitioned into several different classes, the performance of the algorithm depends on the spread of the points surrounding a given test point (as seen in Fig. 2.1).

Fig. 2.1 depicts the case where nearest neighbor representation might perform better than classification-based methods such as [9]. Fig. 2.1(a) divides the domain into several sections, and builds a representative filter at a prespecified location. Fig. 2.1(b) forgoes classification and grabs the k nearest neighbors (in this case $k = 4$). The cluster centroid in Fig. 2.1(a) is not as related to the test point as the four nearest training points in Fig. 2.1(b). The issue for classification-based methods is exacerbated should the initial clustering not provide adequate classes, building the case for nearest neighbor type interpolation mechanisms.

William Freeman's example-based super-resolution [49], which draws from candidate nearest neighbors and chooses the best neighbor by a Markov network, is a good instance of nearest neighbor interpolation. Instead of a single nearest neighbor, which depending on the training set may not be sufficient, manifold learning through neighbor embedding in [28] offers a way to consider k neighbors, incorporating more information to the solution. All nearest neighbor methods have an easier time coming up with local representations, but recent developments [112] show that the underlying assumption in [28] of isometry, at least for Euclidean

[1]Our implementations of all statistical learning methods use large amounts of training data resulting in good quality high-resolution images.


(a) Segmented domain for 2-D low-res patch in classification

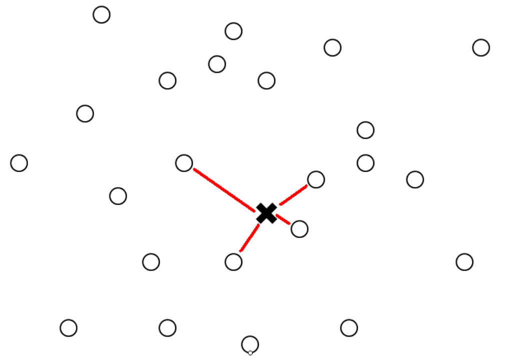
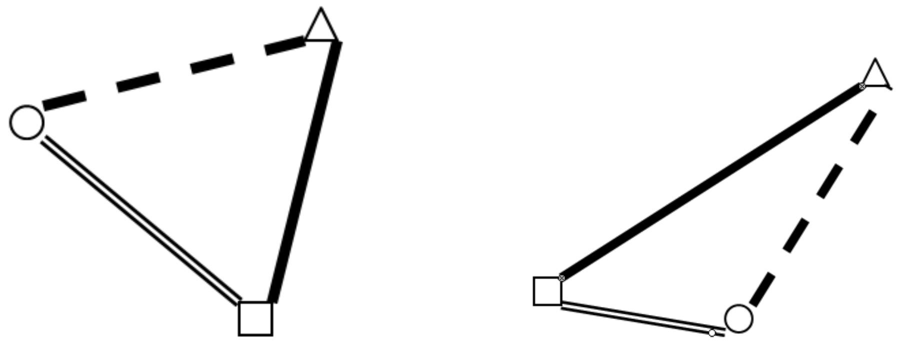

(b) Representation of domain for 2-D low-resolution patch in k-NN

Figure 2.1: Classification versus Nearest Neighbor Representations

distances, between low-resolution neighbors and high-resolution neighbors is inherently false. In other words, the distance metric used by [28] at low-resolution does not correspond equally to the high-resolution distance counterparts. Hence, the weights used at low-resolution are inappropriate for high-resolution construction (seen in Fig. 2.2). Chapter 4 focuses on solving the shortcoming of the non-isometric nature of manifolds and elaborates on issues seen in [9, 49, 28].

To avoid the heavy runtime approximations in k-NN searches, more approximations of parametric estimation approaches are necessary. Such approaches



(a) Euclidean distance in 2-D low-resolution patches (b) Euclidean distance in 2-D high-resolution patches

Figure 2.2: Illustrative Diagram on the Isometry of Image Patches

are usually proposed as combinations of generative and discriminant techniques. As part of our work on kernel-methods, we examine SVR as a learning tool [127] and its appropriately optimized kernel. SVR is part of a broader generalization technique known as support vector machines, a tool based on the idea of structural risk minimization [124]. To be seen in Chapter 6, alternatives to regression by a single SVR include the use of boosting in conjunction with multiple SVR, extensively studied in [43, 93, 69, 61, 75]. The first instance of proposing SVM's for use in a mixture of experts model is in Kwok's Support Vector Mixture [69], in which a hierarchical mixture of SVM's is used in a combination of classification and regression. Parameters in [69] are obtained simultaneously using a single quadratic programming (QP) problem. The complexity involved grows according to both N and C (where N is the number of points and C is the number of classes), and so a simpler, two-step approach involving expectation maximization (EM) is more appropriate. The framework that reflects this reduction follows the example of a notable regression technique in [75]. Chapters 5 and 6 discuss in detail SVR (including use of the DCT domain) and mixtures of SVR techniques for image superresolution.

3 Image Patch-Based Distribution

Mathematics are well and good but nature keeps dragging us around by the nose.

—Albert Einstein

It is wise to explore the input (domain) and output (range) spaces for any given problem. With full knowledge of the domain and range, we are better equipped to model the relationship between them. The goal of this chapter is to explore our input space for image patch-based processing and determine a distribution model that best describes any given representative collection of points from it. That is, for any sufficiently large training set, we find the general form of the multivariate PDF of image patches.

Images and video as inputs are inherently complex entities. For example, Fig. 3.1 contains a multitude of content: letters, water, cloudy sections, sides of buildings, edges, and forestry. Subjective performance of an algorithm on the variety of image and video content focuses on several attributes, including but not limited to temporal and spatial continuity, visual comfort with respect to edges and texture, and clarity of content. Rather than using global image properties (of which there is often little correlation between randomly selected images) or processing an entire image at one time, many techniques focus their efforts on achieving these attributes by observing *local* image properties.

Patch-based algorithms operate locally in exactly this manner and are defined by the processing done in finite dimensional windows consisting of a contiguous subset of pixel locations within a single image. The subset is considered

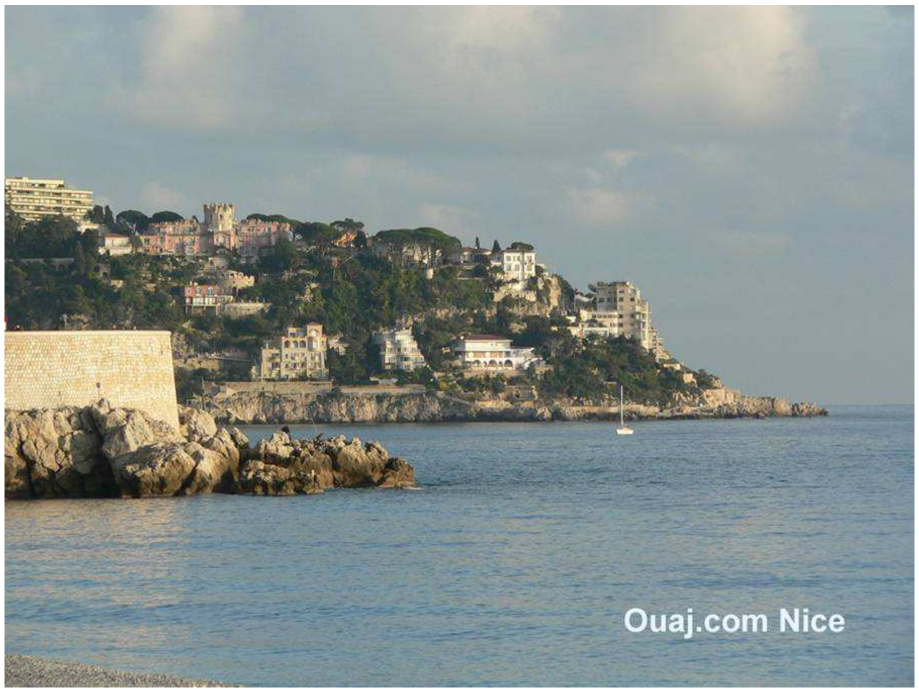


Figure 3.1: Types of Image Content

separately from the rest of the image, which in turn contains additional subsets at different locations that are also processed individually.

Because many image/video compression/processing algorithms find the notion of patch-based processing important (e.g. $N \times N$ DCT compression, motion-estimation, etc., are block-based, while postprocessing techniques such as interpolation are windowed), our study concentrates on the properties of the patches themselves, particularly focusing on their *distribution*.

Anticipating the usefulness of such a study, especially when preprocessing in an input space, several papers [106, 71, 72] have explored the topic. The range of their investigations are diverse, but all of the studies eventually agree on some sort of high-dimensional manifold that cannot be easily explained. [106] assumes two discrete manifolds, termed implicit and explicit, consisting of edges and texture. Likewise, [71] describes image patch space as extremely sparse with data clustered on subspace manifolds, and [72] utilizes synthetic data to offer insight. We explore intuitive explanations behind the manifolds, and why image patches can be broken up as such.

This chapter is divided as follows. Sec. 3.1 proposes a model by considering several related works and some experimental observations. Next, Sec. 3.2 introduces a few tools that aid in explaining the model. Then, Sec. 3.3 provides the necessary parameter estimation steps to use the model. Sec. 3.4 explores the versatility of the model by considering transforms of the input space, and finally in a short note, Sec. 3.5 discusses the dependency of the multivariate PDF on the training set and the need for its cardinality to be finite.

3.1 A Distribution Model for Image Patches

Numerical statistics of image patches depend loosely on the size of the evaluation window or the picture shot depth, which can often be associated with the size of the image itself. For example, 7×7 windows taken from the same sequence at different video sizes (QCIF versus CIF versus HD at $720p$ and $1080p$) have different statistics. The situation is similar to appropriately scaling local statistics to correspond to a camera that zooms in on a particular scene. However, we take the extent of the effect to be minimal, and given a sufficiently large window, [94] suggests approximate scale invariance. That is, if $I(x, y)$ is the observed image, then the statistics of any given patch of I, $patch(I(x, y))$, approximately reflects the statistics of $patch(I(\sigma x, \sigma y))$.

An important measure in data analysis is the *kurtosis*, or normalized fourth order cumulant, of a distribution. When the normalizing value is 3, then the kurtosis of a random variable X becomes the classical metric of the *Gaussianity* of X. Under Gaussian normalization, the definition is given in (3.1).

$$kurt(u) = \frac{\sum_i^N (u_i - \overline{u})^4}{(N-1)\sigma^4} - 3, \tag{3.1}$$

Conventional modeling trends [71] assert that image statistics are seldomly distributed as Gaussians. Instead, any one-dimensional linear mapping of multivariate patch-based distributions, which includes transforms (DCT, DWT, DFT, etc.), display strong *super*-Gaussian characteristics, i.e. $kurt(\mathbf{x}) > 0$ (sharply peaked). One illustration from the extensive collection of domain modeling includes [77], which uses the wavelet detail domain. The histogram is modeled as

$$X \sim \xi_0 e^{-(|h(\mathbf{x})_i|/\alpha)^\beta}. \tag{3.2}$$

for any i where $h(\mathbf{x})_i$ is the i^{th} wavelet detail coefficient. (3.2) results from independently distributed characteristics between detail coefficients, and overall, the proposed model assumes the same in the image patch domain.

The overall form of (3.2) is eventually important to the proposed distribution model, but in terms of the entire multi-dimensional distribution, a single super-Gaussian peak is only partially descriptive. There are limits on the amount of information a single dimension can convey, which is augmented by most models' almost universal zero-mean properties. Alternatively, patched-based applications, e.g. superresolution [9], often model the overall distribution as a mixture of Gaussians (GMM). Rationalization behind such modeling is seldomly justified in any imaging application, the mindset being more default than deliberate.

Nevertheless, as will be further clarified in Sec. 3.2, we believe that the GMM is a surprisingly accurate model in non-smooth areas, and the proposed model combines a GMM with a single super-Gaussian peak similar to [77]. Thus, we propose a multivariate distribution model of the form:

$$X \sim \pi_0 Q\left(-(\mathbf{x}^T\mathbf{x}/\alpha)^\beta\right) + \sum_{i=1}^{C} \pi_i \mathcal{G}(\mathbf{x}, \boldsymbol{\mu}_i, \Sigma_i), \tag{3.3}$$

where $Q(\mathbf{z})$ is some kernel (potentially including a modified Bessel function for multivariate Laplace distributions), $\sum_{i=0}^{N} \pi_i = 1$, and $\mathcal{G}(\mathbf{x}, \boldsymbol{\mu}, \Sigma)$ is a multivariate Gaussian PDF.

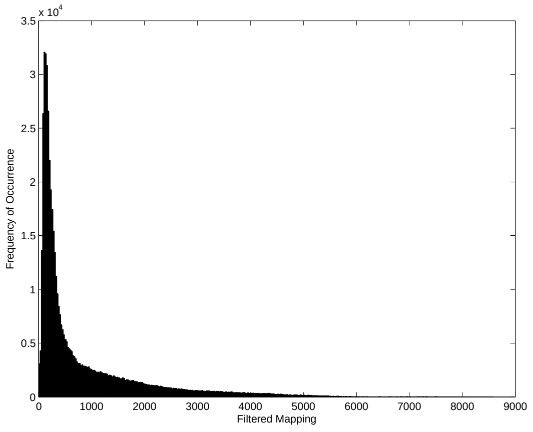
3.2 Empirical Justification

The literature on density estimation under various frameworks is quite extensive. In this section, we observe several statistical frameworks that not only rationalize a parametric method but also verify the functional form of (3.3).

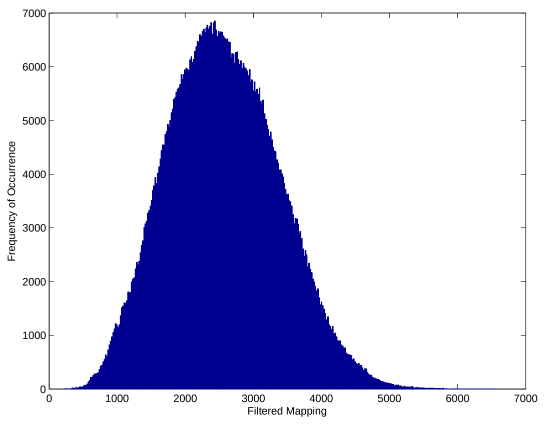
The simplest density estimation technique is the histogram, and we use a mapping as follows for visualization purposes. With $\mathbf{x} = patch\left(I(i,j)\right) \in \mathbb{R}^{d^2}$, a mapping $u \in \mathbb{R}$, is given in (3.4).

$$u = \sum_i |\mathbf{x}_k - \overline{\mathbf{x}}| \tag{3.4}$$

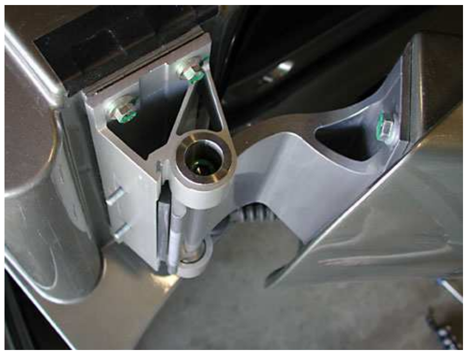
(The mapping is not entirely linear, but is sufficient for our purposes.) [71]'s contention that one-dimensional mappings always have $kurt(u) > 3$ is usually true, as seen in Fig. 3.2(a), where $kurt(u) = 3.203$. There are cases, though, when the image consists of considerable amounts of texture like Fig. 3.2(d), and the histogram will have $kurt(u) \approx 0$ and the Gaussian shape of Fig.3.2(b).



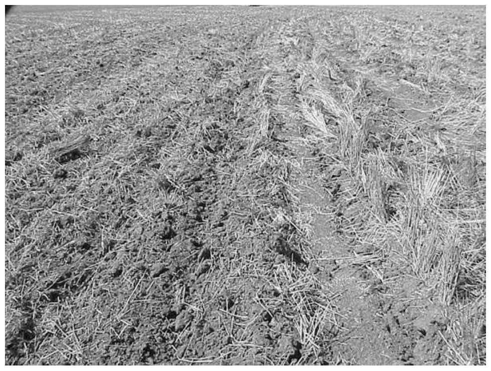
(a) Histogram of Doorhinge



(b) Histogram of Grass



(c) Doorhinge Image



(d) Grass Image

Figure 3.2: Histogram of Multi- to One-dimensional Patch Mapping

Histograms (a,b) of a 1-D mapping from a 7×7 collection of image patches and the corresponding original images (c,d). Trends in (a) and (b) are typical because (c) contains smooth-objects with edges while (d) is mostly texture.

Explanations behind behavior seen in Fig. 3.2 are quite intuitive. The mapping in (3.4) corresponds to a variance-like measure, where small values of u correspond to low variance among pixel values inside a patch $\mathbf{x}$. Low-variance $\mathbf{x}$'s denote smooth image patches with slowly varying gradients, which are abundant in Fig. 3.2(c) and most images in general. Conceptually, they are the image patches that have very low-activity levels, and where $\mathbf{x}$ is mathematically "similar" to $\mathbf{0}$. Such patches appear in high concentrations in most images, where skies, surfaces, solid fabric, etc., are most likely present as large contiguous sets of pixels. Compared to other types of vectors, the large proportion of low-activity vectors form the basis of the first term in (3.3). In fact, the presence of edges is almost unnoticeable in Fig. 3.2(a), with less than a 8% showing. Alternately, Fig. 3.2(b) exhibits potential for parametrization as $kurt(u) = -0.2388$ using the most nonparametric techniques.

The idea is to infer properties of the multivariate distribution from the statistics of the nonlinear projection given in (3.4) by assuming some level of correlation between the two. Given the shape and kurtosis of both histograms, the natural generalization is to fit a super-Gaussian distribution to zero-mean low variance vectors and a GMM to the broad spectrum of texture.

Another traditional distribution modeling technique is k-nearest neighbor (k-NN) [45], where density estimation is mathematically expressed as

$$X \sim \frac{1}{N\sigma} \sum_{i=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{\sigma}\right) \tag{3.5}$$

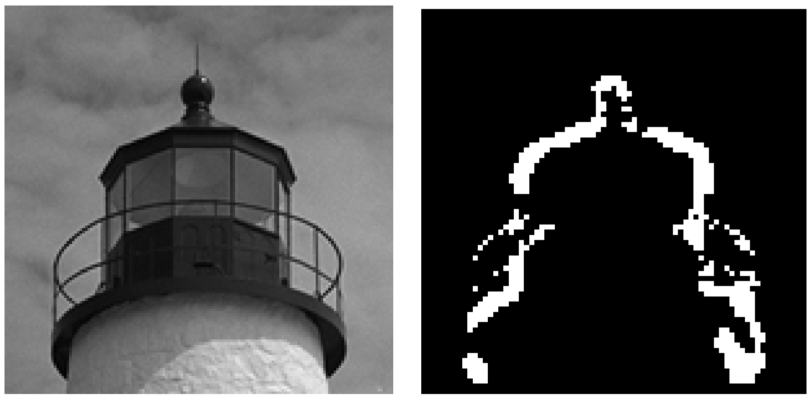
where $K(\mathbf{x}, \boldsymbol{\mu})$ represents a kernel that integrates to one. Conceptually, (3.5) places kernels around each training point for a complete picture of the probability density function (PDF).

The paradigm for our approach is just the opposite, though the final results should remain exactly the same. Instead of thinking that kernels extend around each training point, we can visualize a kernel at the test point since conceptualizing a PDF at runtime is unnecessary. Therefore, distances from the test point to the surrounding training points are evaluated by kernel depending on how far *they* are to the input rather than how far the input is to each of them.

$$k^*(\mathbf{x},\Omega) = \begin{array}{cc} \text{argmin}_k & \sum_{i=1}^{N} W_i(\mathbf{x}_t,\Omega,k)K(\mathbf{x}_i,\mathbf{x}) \geq \eta \\ \text{where} & W_i(\mathbf{x}_t,\Omega,k) \in \{0,1\} \end{array} \tag{3.6}$$

The use of Euclidean distance between nearest neighbors as a goodness of fit measure was introduced in [29], [30], and [44], according to [80]. Using k of the nearest neighbors, (3.6) is essentially a thresholded "distance to nearest neighbor" test. The adaptability of k for common image processing applications will be discussed at length in the next chapter, where the interest is interpolative purposes. In the current context, it is merely a tool in the analysis of our image patch distribution.

Using (3.6), we determine k^* by growing k until η is satisfied by the sum of weights. At some point k-NN becomes ineffective because irrelevant training points begin to be associated with $\mathbf{x}$ in order to meet η. Therefore, we let ζ be a maximum limit on k^*. When $\mathbf{x}$ cannot be represented due to insufficient training, then ζ has been reached in (3.6) or $k^* > \zeta$.



(a) Original Image (b) $|\Omega| = 2 \times 10^6$

Figure 3.3: Example of Patches with Insufficient Training

Fig. 3.3 depicts k-NN findings on the lighthouse image. The original image is Fig. 3.3(a). In Fig. 3.3(b), image patches where $k^* > \zeta$, i.e. insufficient training, are shown in white while well-represented patches are in black.

The conclusion drawn from Fig. 3.3 is that edges are poorly represented by k-NN, and moreover, edge variety is difficult to accommodate in traditional data collection. A training set is, on the other hand, sufficiently representative of smooth patches and texture in a randomly selected image. Observations from (3.3) are intuitive; only a very small minority (albeit the most important parts) of the image can be considered edge patches. Furthermore, variability among edges is extremely high, where orientation, contrast, and sharpness all must be comparable for the patch to be considered similar. Thus, the conjecture that low variance patches and texture form the super-Gaussian and Gaussian mixture of (3.3) resurfaces.

3.3 Using the Proposed Model

Conventional applications of mixture models such as [9] depict each component of the mixture as a class of image content that is independent of other components. Algorithms can provide content-based processing based on the class of the mixture. In practice, the proposed model does not differ much from a GMM, where expectation maximization (EM) experimentation usually yields a number of low-variance, high π_0, and near-zero mean Gaussian.

Yet, by modeling $Q(\mathbf{z})$ differently, higher accuracy may be achieved in the initial term of (3.3). A common super-Gaussian distribution in practice is the multivariate symmetric Laplace distribution, in which it can be shown that $kurt(u) = 3$ for the univariate case. The behavior of multivariate Laplace distributions is attractive because it belongs to a family of *sparse distributions* where there is a high peak at the mean [47]. Ignoring the extraneous terms that are unnecessary for our model, the distribution is defined in (3.7).

$$L(\mathbf{x}, \mathbf{0}, \Sigma) = \frac{2}{((2\pi)^d |\Sigma|)^{\frac{1}{2}}} \left(\frac{\mathbf{x}^T \Sigma^{-1} \mathbf{x}}{2} \right)^{\frac{2-d}{4}} I_{\frac{2-d}{2}} \left(\sqrt{2\mathbf{x}^T \Sigma^{-1} \mathbf{x}} \right), \tag{3.7}$$

where I_λ denotes the modified Bessel function of the third kind.

The parameter Σ must be estimated, and one straightforward estimate [47] involves the sample covariance matrix, which is defined as $\hat{R} = \frac{1}{N}\sum_i(\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T$, where $\hat{\mu}$ is the sample mean. This estimate of Σ is given as

$$\hat{\Sigma} = \frac{1}{|\hat{R}^{\frac{1}{d}}|}\hat{R} \tag{3.8}$$

More advanced parameter estimation techniques include the maximum likelihood estimate, which has been derived in [47]. To make things simple, we use (3.8).

Eventually, parameter estimation for the entire mixture model via EM under (3.3) should give rise to the precise solution. A simpler alternative, as is adopted in our work, would be to first use EM under the assumption of a GMM for the entire domain, then define a hypercube S around $\mathbf{0}$ where training points are relevant to $Q(\mathbf{z})$, and finally approximate parameters of $Q(\mathbf{z})$ using only $\mathbf{x}_i \in S$. In doing so, we replace Gaussian curves in or near S, which may or may not be overfitting the distribution near $\mathbf{0}$, with $L(\mathbf{x})$.

The rationale behind the approximation is that in the initial EM steps, Gaussian curves are placed inside S to accommodate the peaky behavior of the data when in reality, the agglomeration can be attributed to a much simpler phenomenon. Because the Laplace distribution is often described as a multivariate scale mixture of Gaussian models [47], (3.7) serves as an ideal replacement around $\mathbf{0}$.

3.4 Implications in Different Domains

Exploration of multivariate distributions in the spatial domain can be extended to several domains, the analysis of which is exceedingly useful to several applications. For example, the most common issues in video compression deal with image data in the discrete cosine transform (DCT) domain. Knowledge of the distribution of the patches as multivariate vectors could address problems pertaining to compression artifacts, quantization errors, etc. (Note that this is *not* the traditional analysis of the distribution of individual coefficients *within* the patches as independent random variables.) Similarly, detection applications [126] using the discrete wavelet transform (DWT) and the imaging analysis by discrete Fourier transform (DFT) also warrant a study in the versatility of the proposed model.

Linear transforms like the DCT, DWT, and DFT are defined as a correlation-based mapping, where correlation coefficients measure the magnitude of a particular frequency or basis function. Due to linearity, the transforms are merely a change of coordinates. Transforms such as the DCT are designed for energy compaction [99], and as a result, maintain high sparsity. That is, nonzero elements in multi-dimensional vector representation are found in a consistently few number of coefficients. Moreover, the same few coefficients are likely nonzero for any given patch, which is the reason why block-based DCT algorithms use zig-zag scanning [107].

The decorrelation property becomes relevant when evaluating spatial-domain patches in Euclidean two-norm space, i.e. $\mathbb{R}^{d^2 \times 1}$. Because the DCT is a paraunitary transform, Euclidean distance will measure the same quantities before and after the transform [111]. Hence, while the effective dimensionality is reduced from sparsity within patches, the distribution should not appear too different after transformation in the sense that the number, type, and energy of mixture components remain the same. The difference will be the mean, covariance, and alignment of the distribution. That is, the spread and how wide the variance for each component will be altered but not the general form of the distribution. Visually, with $C + 1$ clusters in the spatial domain, the tails of the $C + 1$ corresponding clusters in transform domain will only extend in a few directions due to sparsity within patches. By examining the DCT, the following proof can be extended to the DFT and DWT domains.

Claim: Given the model as defined in (3.3) with the spatial domain image patch $\mathbf{x}$, the DCT of $\mathbf{x}$ will be distributed according to the same mixture model (of the form (3.3)) only with different mean and covariance parameters.

Proof: Let $P \in \mathbb{R}^{d \times d}$ be an image patch or block in matrix form. The two-dimensional DCT is a separable transform that is traditionally written as

$$DCT_{2D}(P) = C^T P C \tag{3.9}$$

where C is a matrix defined by

$$C = k_m \begin{bmatrix} 1 & cos(\frac{1}{2}\frac{\pi}{N}) & cos(1\frac{\pi}{N}) & \cdots \\ 1 & cos(\frac{3}{2}\frac{\pi}{N}) & cos(3\frac{\pi}{N}) & \cdots \\ 1 & cos(\frac{5}{2}\frac{\pi}{N}) & cos(5\frac{\pi}{N}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3.10}$$

To evaluate the multivariate distribution, we must vectorize P by appending the columns into a single columned arrangement, which we denote by $\mathbf{x}$. Therefore, $\mathbf{x} \in \mathbb{R}^{d^2 \times 1}$ denotes an image patch for which we have already defined the distribution in (3.3). We can now define a matrix A as

$$A = \begin{bmatrix} C_{1,1}C & C_{1,2}C & C_{1,3}C & \cdots \\ C_{2,1}C & C_{2,2}C & C_{2,3}C & \vdots \\ \vdots & \vdots & \ddots & \\ C_{N,1}C & C_{N,2} & \cdots & C_{N,N}C \end{bmatrix} \tag{3.11}$$

where $C_{i,j}$ denotes the $(i,j)^{th}$ element in C as defined in (3.10). Equivalent operations for $\mathbf{x}$ can now be written from the matrix operations of P in (3.9), and the two-dimensional DCT in vector form can be written as

$$\mathbf{y} = vect\left(C^T P C\right) = A\mathbf{x} \tag{3.12}$$

Because $Y = AX$ is an invertible function of $\mathbf{x}$, where $X = A^{-1}Y$ (and by orthonormality, $X = A^T Y$), the probability distribution function (PDF) of the random vector $\mathbf{y}$, $f_Y(\mathbf{y})$, is written as

$$\begin{aligned} f_Y(\mathbf{y}) &= \sum_i \frac{1}{|\mathcal{J}(\mathbf{x}_i)|} f_X(\mathbf{x}_i) \\ &= \frac{1}{|\mathcal{J}(\mathbf{x})|} f_X(A^{-1}\mathbf{y}) \end{aligned} \tag{3.13}$$

where the Jacobian, $\mathcal{J}(\mathbf{x})$ is equal to A^T.

Because (3.3) is a linear combination of terms and the Jacobian in (3.13) is of a linear function, individual terms will map one to one. Therefore, we simply need to ensure that each component of (3.3) after transformation remains in the same form as the original, spatial domain mixture in order to show that the framework is indeed the same only with different parameters.

Beginning with the second term in (3.3), the GMM, let X_i be the i^{th} Gaussian in the mixture of (3.3), such that

$$X_i \sim \left|(2\pi)^d \Sigma\right|^{-\frac{1}{2}} \exp\left\{\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_i\right)^T \Sigma^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_i\right)\right\} \tag{3.14}$$

Then, there exists an Y_i, which we define as the i^{th} mixture component, in the resultant transformed mixture model such that $Y_i = AX_i$. Using (3.13) and (3.14), the mixture component can be written as

$$Y_i \sim \frac{1}{|A^T|}\mathcal{G}\left(A^{-1}\mathbf{y}, \mu, \Sigma\right) \tag{3.15}$$

where it can easily be shown that Y_i has the form

$$Y_i \sim |2\pi A^T \Sigma A|^{-\frac{1}{2}} e^{\left\{\frac{1}{2}(\mathbf{y}-A\boldsymbol{\mu}_i)^T \left(A^T\Sigma A\right)^{-1}(\mathbf{y}-A\boldsymbol{\mu}_i)\right\}} \tag{3.16}$$

which is a Gaussian with mean $A\boldsymbol{\mu}$ and covariance matrix $A^T\Sigma A$.

Finding the form of the Laplace distribution requires more thought. Let X_0 denote the random variable representing the first component of (3.3), the Laplace distribution component in the spatial domain, and let Y_0 denote the corresponding component after transform. The characteristic function of the general multivariate symmetric Laplace distribution [67] is given by:

$$\Psi_{X_0}(\boldsymbol{\omega}) = \frac{1}{1+\boldsymbol{\omega}^T\Sigma\boldsymbol{\omega}} \tag{3.17}$$

The affine property of Fourier transforms (by extending the bivariate case in [20] to the multivariate case) states that, given a linear transformation A in $\mathbb{R}^{d\times d}$, the corresponding Fourier transform, denoted as $\mathcal{F}(\boldsymbol{\omega})$, of the multi-dimensional vector $\mathbf{x}$ is

$$\mathcal{F}(A\boldsymbol{\omega}+\mathbf{b}) = \frac{1}{|A^T|}e^{j(2\pi/|\Sigma|)\boldsymbol{\omega}^T\mathbf{b}}\mathcal{F}(A\boldsymbol{\omega}) \tag{3.18}$$

Because $\mathbf{b} = \mathbf{0}$, this could have been derived from (3.13) easily.

$$\Psi_{Y_0}(\boldsymbol{\omega}) = \frac{1}{|A^T|}\mathcal{F}\left(f_{X_0}(A^{-1}\mathbf{y})\right) \tag{3.19}$$

We can integrate the initial term into the scaling factor of (3.7) to write the final characteristic function of Y_0:

$$\Psi_{Y_0}(\boldsymbol{\omega}) = \frac{1}{1 + \boldsymbol{\omega}^T A^T \Sigma A \boldsymbol{\omega}} \tag{3.20}$$

which upon inspection is the characteristic function of a multivariate Laplace distribution with covariance matrix parameter $A^T \Sigma A$. Thus, the two-dimensional DCT of a zero-mean vectorized patch with a multivariate Laplace distribution is also a multivariate Laplace distribution, only with a different scale factor.

Therefore, because each of the components of the mixture model satisfies either (3.16) or (3.20), then we have shown that Y is of the same form as (3.3).

3.5 On the Variability of Finite Data Sets

An underlying assumption behind (3.3) is that the data set from which parameters are drawn is finite. Predicting the framework for the distribution of the true nature of all possible image patches is unrealistic. An infinite amount of data including all possible image content may result in an infinite amount of mixture components, where small variations that can possibly exist in texture equate to a different mixture.

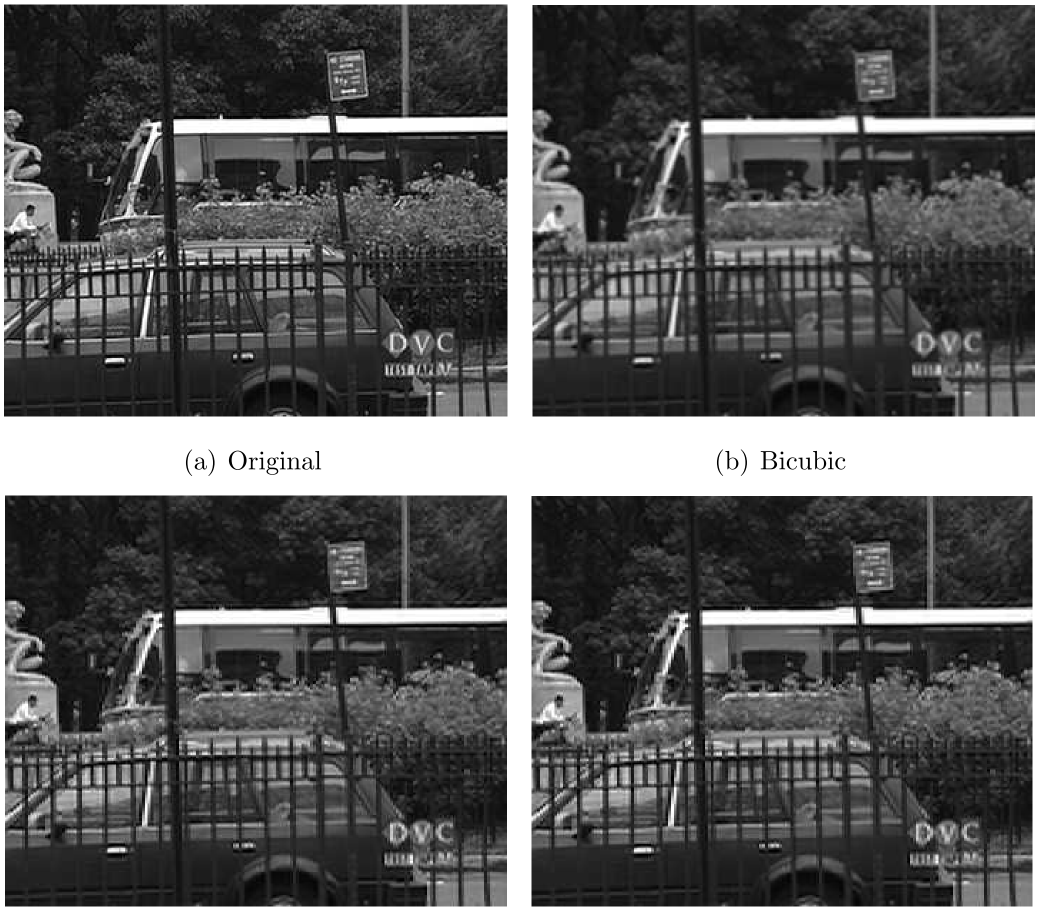
However, given a finite training set size, we can accurately predict how the cohesive elements inside should behave independently of each other. Therefore, rather than predicting the distribution of image patches in general, it is important to note that the distribution of image patches is of a given data set, and that though the location, number of mixture components, and scale parameters may differ, the underlying framework remains the same.

3.6 Results

There are a multitude of standard evaluation tools to determine how accurately a statistical model reflects naturally occurring phenomena. We assess the quality of the proposed model through its performance in an application.

A number of algorithms assume parametric models in image processing. One of the simpler applications that illustrates the effectiveness of the proposed model is content-based image interpolation, where [9] is the comparison. Given an image patch $\mathbf{x}$, [9] relies on a GMM to classify image content,

Following Sec. 3.3 and using an S with side value 10, we redistribute the π_i values of the eliminated Gaussians using volumes under the curves, coming up with a new $\boldsymbol{\pi}$ as consistent with (3.3). The results for the GMM interpolation and (3.3) along with bicubic interpolation are shown in Table 3.1 and Fig. 3.4. Table 3.1 refer to SQCIF, QCIF, and $720p$ formats.



(a) Original

(b) Bicubic

(c) GMM Only

(d) Laplacian-GMM

Figure 3.4: Visual Results of Classification-Based Approaches

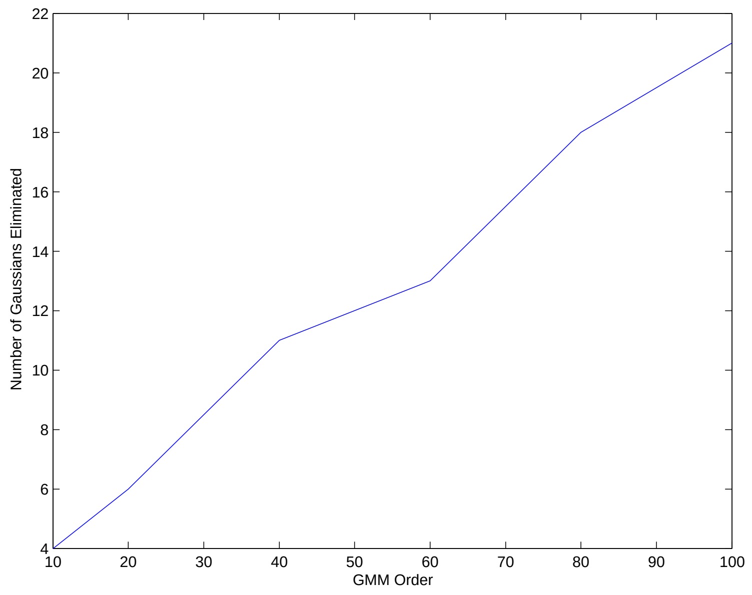


Figure 3.5: Properties of Reduction in Model Order

The contention with respect to interpolation is that disparity between the reconstruction efforts of GMM and (3.3) is negligible. Moreover, as the difference in operation involves vectors near **0**, meaning there is little or no variance in pixel values within the vectored patch, visual differences are imperceptible. The conclusion in Fig. 3.4 is that (3.3) with 15 components achieves the same performance as a GMM of order 20, where a submixture of six Gaussians has been eliminated.

Table 3.1: Comparisons of Interpolation Model Replacement

	Low-Res	Med-Res	High-Res
GMM-Based	22.6531 dB	22.9722 dB	20.0156 dB
Model in (3.3)	22.6529 dB	22.9722 dB	20.0156 dB

Consistent clustering around **0** occurs quite frequently, and under different training sets, the behavior remains the same. Fig. 3.5 demonstrates the near linear trend of reduction in model components of (3.3) as a function of initial GMM components. In replacing the Gaussians, (3.3) is validated because a mixture of Gaussians with close means is successfully modeled as a Laplace distribution.

Generalizing results seen in Fig. 3.4 to trends in Fig. 3.5, (3.3) offers a simpler model than a given GMM with many mixture components around $\mathbf{0}$.

3.7 Summary

A unifying model of image patches for general use has been proposed. The model is based on observations that images usually consist of a large proportion of low variance image content. Parameters are approximated using the data points in a prespecified hypercube, not the entire training set. Except for around $\mathbf{0}$, a GMM accurately depicts a training set of image patches. Around $\mathbf{0}$, we require a super-Gaussian shaped distribution with $kurt(\mathbf{x}) \approx 3$, which a multivariate Laplace distribution will supply. Therefore, the final distribution has a Laplace + GMM framework. Applying linear transforms such as the DCT, DFT, and DWT alter the parameters of the mixture model, but do not alter the general framework of the PDF. The model has been tested in a conventional interpolation framework and is shown to be equivalent in quality while simpler in model complexity than the GMM.

3.8 Acknowledgements

Portions of Chapter 3 appear in "A Model for Image Patch-Based Algorithms", Karl Ni and Truong Nguyen, to appear in presentation at the *IEEE International Conference on Image Processing*, October 2008. The dissertation author was the primary author of the publication, and the co-author listed directed and supervised the research which forms the basis for this chapter.

4 Adaptive k-Nearest Neighbor for Image Interpolation

When in doubt, use brute force.

—Ken Thompson

Introduced in Chapter 2 are the static interpolative splines methods [57, 119, 81, 66]. Splines operate on the incorrect assumption (which has been widely acknowledged as incorrect) that relationships between local low and high-resolution content can be described by a single convolutional kernel. Rather, their complexity usually justifies their usage, and though numerical results are most likely incorrect, the visual quality does not overtly reflect it. More generally, the HVS is forgiving of estimation errors from reasonable linear filters, meaning that in terms of human perception, the actual process is *approximated* fairly well. The observation is fundamental to the proposed algorithm in this chapter where we propose content-specific linear filters as the regression technique of choice. Thus, when errors do occur, the damage of estimation errors due to insufficient training appears perceptually mitigated.

Instead of a static interpolation process for all image content, a logical improvement [129, 74, 3, 9, 24] would be to adapt filters that are optimized for certain content. Such algorithms balance specificity with estimation errors by benefitting from the knowledge that local linear filtering will not significantly distort the image. We are interested in adapting optimal filters that are chosen from a training set by variants on the k-Nearest Neighbor (k-NN) algorithm.

k-NN determines k training points that are closely related to an input vector through an appropriate similarity metric. For image interpolation, once the relevant training samples are found, filters are specially tailored to determine high-resolution values after identifying low-resolution content. The crux is to achieve specificity with regard to image content without any loss of generalization of application. That is, how detailed can we make an image look while still maintaining a broad base of applicability?

The answer to this question is intimately related to the number of training samples used per reconstruction filter. In images, more training points per filter, i.e. k is large, equals more generality, meaning that errors and variations due to the training set are diminished. Likewise, fewer training points per filter, i.e. k is small, equals more specificity, meaning that the image reconstruction is clearer and more detailed. As will be explained in this chapter, variables naturally depend on the size and quality of the overall training set, but it is reasonable to conclude that to accommodate any possible test input, k must be variable.

4.1 Review of k-Nearest Neighbor for Regression

The k-nearest neighbor [50] rule is among the simplest statistical learning tools in density estimation, classification, and regression. Trivial to train and easy to code, the non-parametric algorithm is surprisingly competitive and fairly robust to errors given good cross-validation practices.

Let Ω be a training set of N input-output pairs. Then,

$$\Omega = \{(\mathbf{x}_1, \mathbf{y}), (\mathbf{x}_2, \mathbf{y}_2), \cdots, (\mathbf{x}_N, \mathbf{y}_N)\} \tag{4.1}$$

For the problem specifically relating to image interpolation, $\mathbf{x}_i$ is comprised of the i^{th} low-resolution image patch in the training set. Likewise, $\mathbf{y}_i$ defines the i^{th} high-resolution image patch. During runtime where we denote values with the subscript t, the adaptation of k-NN determines the high-resolution image patch $\mathbf{y}_t$ from a single low-resolution image patch $\mathbf{x}_t$. For mathematical reasons, it is easier to represent image patches $\mathbf{x}$ and $\mathbf{y}$ as vectors instead of square patches. Therefore, in subsequent derivations $\mathbf{x}$ and $\mathbf{y}$ are both vectors that have been rearranged from image blocks into a single column.

The typical k-NN estimate for regression [40, 41] at the test point $\mathbf{x}_t$ is given as:

$$\hat{\mathbf{y}} = g(\mathbf{x}_t) = \frac{1}{k}\sum_{i=0}^{N} W_i(\mathbf{x}, \Omega)\mathbf{y}_i \tag{4.2}$$

where $W_i \in \{0, 1\}$ depending on whether or not $\mathbf{x}_i$ is among the k nearest neighbors of $\mathbf{x}_t$. (To be seen in later sections, instead of g, a linear transformation G performs the task of $\mathbf{y}_t = G\mathbf{x}_t$.)

Naturally, the definition of (4.2) can be extended by not necessarily limiting W_i to 0 or 1, but rather only the constraint $\sum_{i=1}^{N} W_i = k$. In fact, there are several common weighting schemes, ranging from posterior probability like expressions [45] to iteratively determined convex solutions [7], all functions of distances or weights that can used to minimize some criterion as in [28].

An extensive study on error rates for regression-based k-NN estimates was analyzed in [34], where it was conceded that for (x, y) jointly normal[1], under the squared-error loss case, the unconditional, large sample risk R as $N \to \infty$ of the k-NN estimate satisfies

$$R_N^{(k)} = \left(1 + 1/k + \frac{\sigma_1^2}{\sigma_2^2} E\left[x_t - \frac{1}{k}\sum_{i=1}^{k} x_{i,N}\right]^2\right) R^* \tag{4.3}$$

Here, R^* is the Bayes risk (minimum expected loss), $x_i \in \Omega$, and parameters σ_1 and σ_2 are variance parameters in probability distribution functions (PDF's) $f(y)$ and $f(y|x)$. (4.3) trades off large k values to keep the $1/k$ term small while simultaneously favoring a smaller k to keep the final term in (4.3) small. This type of tradeoff is common in k-NN problems, and there is need for cross-validation even for our proposed adaptable k values.

N is not always large, and for such cases, the risk R_{NN} of nearest neighbor (where $k = 1$) is usually smaller than the risk of k-NN [34], but overall, when N is large, k-NN is invariably the rule of choice. Actually, when N is large and the

[1] [34] only treats the univariate case.

dimensionality d is small, k-NN is almost always preferable or at least competitive among other estimation techniques such as SVR. (SVR performs well when N is small and d is large.[2]) The intuition is to blanket the entire domain with samples, possible only with large N and sufficiently small d.

It is latter scenario that tends to be the case with the potential of today's computing power. While memory for computing tasks increases and computing time for higher complexity routines decreases, potential to support large N values motivates the application of k-NN to various problems (not limited to interpolation and superresolution.) The following sections offer a basic but original k-NN algorithm while referencing existing k-NN approaches. It may also be worth it to review Chapter 3, where we have introduced the idea that a kernel can extend out from a given test input instead of placing kernels at every training point. The paradigm with respect to finding the optimal k is the same as Chapter 3, and we elaborate in throughout the rest of this chapter.

4.2 Building the Optimal Interpolation Mechanism

To reiterate, the kernel function K used in the proposed algorithm is the RBF, given in (4.4).

$$K_{\mathbb{F}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2\pi\|\Sigma\|} \exp\left\{d_{\mathbb{F}}(\mathbf{x}_i, \mathbf{x}_j)\right\} \leq 1 \tag{4.4}$$

where $d_{\mathbb{F}}(\mathbf{x}_i, \mathbf{x}_j)$ is the Mahalanobis distance or weighted Euclidean distance specified by $\frac{1}{2}\left(\mathbf{x}_i - \mathbf{x}_j\right)^T \Sigma^{-1} \left(\mathbf{x}_i - \mathbf{x}_j\right)$. Unfortunately, in the absence of prior knowledge, most k-NN algorithms determine proximity through un-weighted Euclidean distances. We can calculate the Σ of the entire training set and use a scaled version of it, which does not actually offer a significant improvement.

There are many weighting schemes for W in (4.2) as alluded to in Sec. 4.1. One general family of solutions, known as *locally weighted regression* (LWR) [31, 8],

[2]SVR is more of a generalizing technique.

replaces W by a particular model class $g(\mathbf{x}, \boldsymbol{\beta})$, in which $\mathbf{y}_t$ is determined locally by functions based on how similar point $\mathbf{x}_i$ is to $\mathbf{x}_t$ [105]. Then, the task of k-NN for regression falls to estimating select parameters for reconstruction in (4.5).

$$\boldsymbol{\beta}^* = \operatorname{argmin}_\beta \sum_{\mathbf{x}_i \in \text{neighborhood}} d_{\mathbb{R}}\left(g(\mathbf{x}_i, \boldsymbol{\beta}), \mathbf{y}_i\right) K\left(d_{\mathbb{F}}(\mathbf{x}_i, \mathbf{x}_t)\right) \tag{4.5}$$

where $d_{\mathbb{R}}$ and $d_{\mathbb{F}}$ are distance metrics in the range and feature space, respectively.

On the principle that isometry is not a realistic scenario for image superresolution [112], (4.5) becomes a viable alternative. We require a single assumption with LWR that linear filtering yields an excellent approximation for local image construction as opposed to assuming some kind of duality between low-resolution and high-resolution manifolds in [28]. Hence, $g(\mathbf{x}, \boldsymbol{\beta})$ in (4.5) becomes the linear filter in question, which can be reduced to an MMSE filter formulation. We can find $\mathbf{y}_t$ by

$$E\left[\mathbf{y}_t | \mathbf{x}_t\right] \approx g(\mathbf{x}_t, \boldsymbol{\beta}) = G\mathbf{x}_t \tag{4.6}$$

where G is a $u \times d$ matrix, u being the upscaling factor, and is constructed by probability parameters β and neighboring low-resolution and high-resolution pairs.

(4.6) focuses on determining G, which is accomplished by slightly modifying traditional MMSE equations. Eventually, pre-processing steps such as mean-shifting or variance normalization are implemented to determine the feature space $\mathbb{F}$ for both k-NN neighbor identification and regression, but for now, the filters are created in a manner similar to [9]. To manage the data, let us assemble the test vector neighbor pairs of low-resolution vectors $\mathbf{x}_i$ and high-resolution $\mathbf{y}_i$ into X and Y matrices, respectively. The arrangement of X and Y matrices is as follows:

$$X = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_1 & \mathbf{x}_2 & \vdots & \mathbf{x}_i & \vdots & \mathbf{x}_N \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

and

$$Y = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{y}_1 & \mathbf{y}_2 & \vdots & \mathbf{y}_i & \vdots & \mathbf{y}_N \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{4.7}$$

By defining (4.4), we can construct a matrix P for a given neighborhood of $\mathbf{x}_t$ such that if $\mathbf{p}$ is a vector of similarity measures whose i^{th} entry is the value $K_{\mathbb{F}}(\mathbf{x}_i, \mathbf{x}_t)$, then

$$P = \mathbf{1}^T \mathbf{p} \tag{4.8}$$

where $\mathbf{1}$ is a k dimensional vector of all ones. Hence, P has dimension $k \times p$.

The purpose of P is to establish a proper weighting of point $\mathbf{x}_i \in \mathcal{N}(\mathbf{x}_t)$. Since one of the arguments to the $K_{\mathbb{F}}(\mathbf{x}_i, \mathbf{x}_j)$ in (4.4) is always $\mathbf{x}_t$, weighting schemes [7] usually observe the similarity between $\mathbf{x}_i$ given $\mathbf{x}_t$ as a Gaussian PDF with mean centered at $\mathbf{x}_t$ and the elements in P as how probable that neighborhood vector is relevant.

The sample autocorrelation matrix $R_{XX} = XX^T$ and cross-correlation matrix $R_{XY} = XY^T$ lead to the final expression for G in (4.9), which is roughly equivalent to the derivations from [9].

$$G = (R_{XY} \cdot P)\,(R_{XX} \cdot P)^{-1} \tag{4.9}$$

4.2.1 Choosing the Correct k

Top among the attractive properties of linear regression is the robustness to poorly designed filters in image processing. The explanation stems from their tendency to average out image data. Moreover, weighted MMSE solutions can be interpreted as an averaging of an overdetermined system. When more and more points are involved, or k is grown to be large, the sample size of X grows as well, and G will be able to accommodate a more general base of $\mathbf{x}_t$ vectors. On the other hand, maintaining large k defeats the purpose of k-NN because using smaller k ensures specialization of the resulting filter.

The idea of choosing an adaptable k compensates for non-uniformly distributed training data when the high dimensional domain of low-resolution image patches is inconsistently scattered across the feature space. For the uniformly distributed case, k can be fixed because the variance in the distances from the surrounding training points to any given input test point is limited among all possible test points. In such cases, using a fixed k will always yield roughly equivalent relevancy in training information among any two test points. As it happens, in natural image statistics, the entropy of image patches is difficult to describe and cannot be modeled in any traditional linear sense. The investigation of image patch statistics has been extensively examined in several papers such as [72, 106][3] and in Chapter 3, where patches are discovered to be sparsely distributed with the majority of points clustered in high concentration on non-linear manifolds.

The view of clustered high-dimensional manifolds is ingrained in most image model descriptions in some form, and work on the subject is abundant [72, 106, 58, 27]. We are not concerned with the specific properties, merely exploiting the fact that they exist. Thus, the proposed algorithm should be tuned and adaptable to the distribution and where arbitrary test inputs might land. The manifestation of such an endeavor is intimately connected to the number of nearest neighbors, or k, given the input test point in a naturally distributed training set.

The goal is to find the right k for a desired tradeoff. As one may guess, $k < k^*$ overfits the training set, specializing G too much, and the manifestation is a grainy and discontinuous image. Furthermore, if k were exceedingly small, $k \ll k^*$, G could become non-singular. The intuition is that training points near $\mathbf{x}_t$ could be very close together causing (4.9) to be underdetermined. Analytically speaking, vectors in X that are too similar can mean that R_{XX} is rank deficient and thus non-invertible. This is a dilemma because when k-NN finds the most relevant data with respect to $\mathbf{x}_t$, the collected vectors based on $\mathbf{x}_t$ are similar to each other as well. Hence, though it is counterintuitive, it is important to choose a large enough neighborhood in $\mathbb{F}$ so that diversity in the $\mathcal{N}(\mathbf{x})$ exists.

Built-in error is a consequence of uneven training data collection in unsupervised learning. Similarly, uneven training data also has implications that k^* may potentially be significantly different for any two given test points. The optimal k^* has been initially given in (3.6) in Chapter 3, and we reiterate the equation

[3] [106] assumes two manifolds termed "explicit" and "implicit", but the two types effectively describe the same behavior as [72]

here:

$$\begin{aligned} k^*(\mathbf{x}, \Omega) = \operatorname{argmin}_k \quad & \sum_{i=1}^{N} W_i(\mathbf{x}_t, \Omega, k) K(\mathbf{x}_i, \mathbf{x}) \geq \eta \\ \text{where} \qquad & W_i(\mathbf{x}_t, \Omega, k) \in \{0, 1\} \end{aligned} \tag{4.10}$$

The expression in (3.6) obtains k^* by finding the minimum number of neighbors whose sum of similarity measures exceeds a threshold η, which is obtained through cross-validation. Moreover, η is a minimum bound of k since $K(\mathbf{x}_i, \mathbf{x}_t) \leq 1$ for all $\mathbf{x}_i$.

Analyzing (3.6) for a given $\mathbf{x}_t$, if there are only a few $\mathbf{x}_i$ with high probability of being related to it, that is $\sum_i K(\mathbf{x}_i, \mathbf{x}_t)$ is small, then the proposed algorithm will need to consider more points in hopes of generalizing well. Alternatively, if there are many $\mathbf{x}_i$ that are related to $\mathbf{x}_t$, i.e. $\sum_i K(\mathbf{x}_i, \mathbf{x}_t)$ is large, it is unnecessary to use other points where the similarity is low because the specialized filter generated by the points within $\sum_i K(\mathbf{x}_i, \mathbf{x}_t) \leq \eta$ is very likely to be accurate. Conceptually, we can visualize a ring that extends further and further depending on whether or not there are enough points inside the ring.

4.3 Heuristics for Insufficient Training

k-NN algorithms assume there are enough points to blanket the entire domain, providing a good density estimate of the input space. Problems arise from insufficient training because the further the ring of values under consideration extends, the smaller the similarity values, and the less suited any additional training point is to complete the task of reaching η. In extreme cases, η may not even be reached before the entire training set is exhausted of points. Thus, we require the incorporation of a simple heuristic of limiting the maximum value of k that is allowed to be used.

In setting a limit on k^*, $\zeta = k_{max}$, a generic technique, i.e. bicubic interpolation, may be used for those $\mathbf{x}_t$ that Ω does not represent well. Additionally, the maximum number of neighbors will reduce both complexity and errors. The complexity reduction should be obvious, but to see that errors have been minimized by stopping the algorithm prematurely with ζ, the algorithm effectively acknowledges that, at least for the image patch at hand, the original intention of the proposed algorithm cannot be carried out due to a less than competent training set. Therefore, for any $\mathbf{x}_t$ that k-NN is ill-equipped to manage (i.e. $k^* > \zeta$), the errors are bounded by whatever interpolation algorithm replaces k-NN.

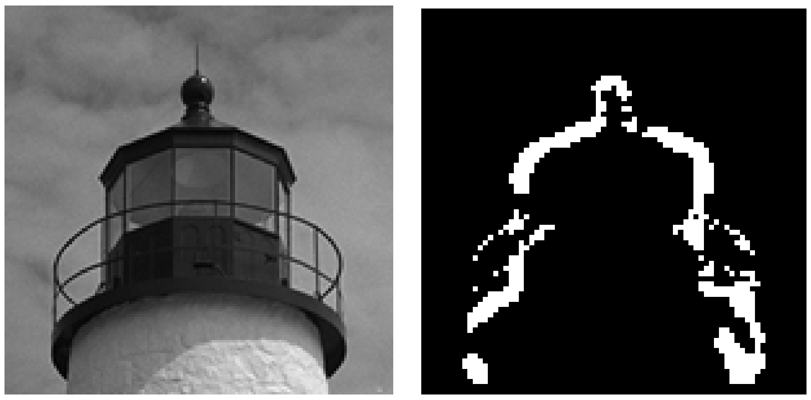
The question now becomes finding what kind of interpolation algorithm should replace k-NN. Is there a particular type of image patch that the k-NN algorithm consistently disfavors? Moreover, based on this bias, are there certain properties of these patches that allow us to tailor a solution using this knowledge? The answer is yes on both accounts. After running several tests, we came across a peculiar reoccurring theme in generic training and testing images: texture patches never reached ζ and appeared at high quality, but edge patches often did and needed attention.

Using 2×10^5 training points and observing similarity measures in (4.8) (which are based on Euclidean distances), the texture matches usually retain similarity values of $K(\mathbf{x}_t, \mathbf{x}_i) \approx 0.93$ (out of 1.00) , whereas edge matches usually satisfy $K(\mathbf{x}_t, \mathbf{x}_i) \leq 0.40$. Furthermore, in viewing a single image,[4] only a small percentage of image patches are actually edges, so accumulating relevant image patches in (3.6) to surpass η is even more improbable. The situation is best described in Fig. 4.1, which was originally shown in Chapter 3.

With only 200 thousand data points, we cannot actively reconstruct many edges on the lighthouse because training patches don't occur frequently enough and there aren't close enough matches. Texture, however, *can* be, and much of the texture interpolation occurs because $k^* < k_{max}$.

Though texture results in high peak signal to noise ratios (PSNR), to be presented in Sec. 4.6, the human visual system (HVS) focuses on edges [22]. Fortunately, research into edge-oriented image filtering has been well-studied [114, 52]. In our framework, we can agglomerate a bank of edge-oriented filters that do "well-enough" when the "best" filter through k-NN is unavailable, effectively reducing

[4]Unsupervised data collection means that the exact percentage values of patches are not known.



(a) Original Image (b) $|\Omega| = 2 \times 10^6$

Figure 4.1: Example of Patches with Insufficient Training

the implementation to a specialized version of [9] with an added MRF improvement (see the next section, Sec. 4.4) through [73].

With enough data points, however, replacing k-NN conditioned on $k^* > \zeta$ should occur relatively few times. That is, edges may and often are well-represented in the training set, which indicates the algorithm is operating closer to capacity. Studies such as these are reserved for current and ongoing work.

4.4 Using the Single-Pass Approximation to the Markov Random Field (MRF)

Among the nearest neighbor literature used for image interpolation, [49] most closely resembles the proposed algorithm. The differences between the proposed algorithm and [49] are subtle but significant. [49] chooses a single neighbor among a number of candidate neighbors and applies the high-resolution differences. Our algorithm chooses an "appropriate" quantity of candidate neighbors and directly determines the high-resolution content. The philosophy arises from differences in the choice of feature space, where [49] first uses an analytic interpolation scheme, such as bicubic interpolation, and stores differences between the initial interpolation and the true high resolution patch. In contrast, the preprocessing involved in our algorithm only consists of DC subtraction and scaling. We trust the capacity of the *algorithm* to perform the approximation rather than first approximating outside the algorithm.

Despite the difference, [49] makes a good argument that globalization in terms of relating neighboring patches is necessary. Consequently, we have followed their lead by considering the usage of a Markov network in modeling spatial relationships between patches. Markov modeling techniques usually require the use of some annealing process, which is usually computationally intractable for most interpolation purposes. Therefore, with the aid of [73], we have implemented a simpler-than-MRF, single-pass technique to enhance coherency from patch to patch.

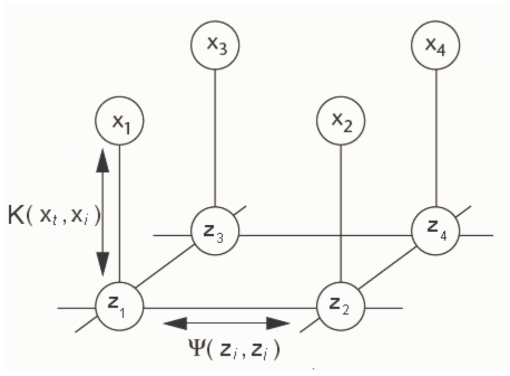


Figure 4.2: Markov Random Field Implementation

Fig. 4.2, with the exception of terminology, is very similar to the one in [49]. We use $K(\mathbf{x}_t, \mathbf{x}_i)$, where $\mathbf{x}_t$ is the observation and $\mathbf{x}_i \in \Omega$ in Sec. 4.2 to determine possible states, $\{\mathbf{z}\}$. We can use a function similar to K for Ψ to determine the compatibility between the states.

Single pass algorithms include extra arguments into the decision making process that increase propensity towards one neighbor over another. Because our algorithm observes multiple neighbors per input patch, the structure of the one pass algorithm must be modified somewhat.

Given the filter construction process in (4.9), we can take advantage of an expression that is already designed to penalize or reward training points through a matrix P. To review, elements within P denote the importance of a particular training point. After determining the k^* values for all image patches (or realistically, just the ones surrounding the test patch being evaluated), the logical course of action would be to reward those states that contain high values for $K(\mathbf{x}_i, \mathbf{x}_j)$ and $\Psi\left(\mathbf{z}^{(x_1,y_1)}, \mathbf{z}^{(x_2,y_2)}\right)$. Adding a scaling factor α, a very simple conditioning scheme could be

$$P_{(i,\cdot)} = K(\mathbf{x}_t, \mathbf{x}_i) + \alpha \sum_{n \in \mathcal{N}} \sum_{j} \Psi\left(\mathbf{z}_j^{(n)}, \mathbf{z}_i\right) \tag{4.11}$$

Here, $\mathbf{z}_j^{(n)}$ refers to the j^{th} candidate state (see Fig. 4.2) of the n^{th} low-resolution (LR) block in $\mathcal{N}$, the neighborhood of the input block. Referring to the entire algorithm in Fig. 4.4, blocks in $\mathcal{N}$ are adjacent to the input test block.

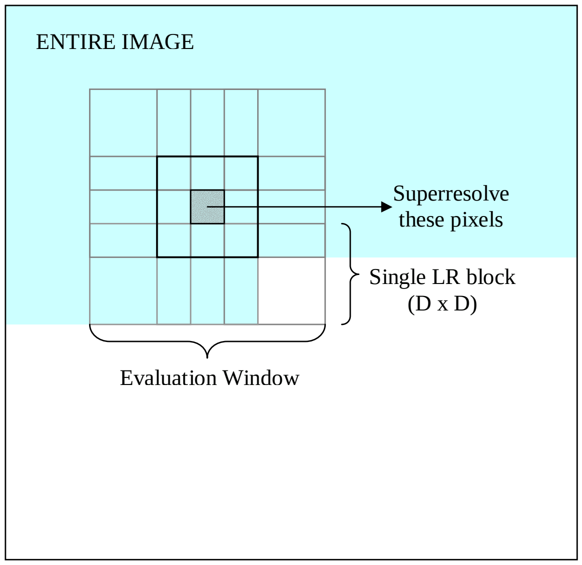


Figure 4.3: Description of Windows and Blocks for Adaptable k-NN

Fig. 4.3 presents the evaluation window, in which all k neighbors are known or need to be found. The framework of the entire algorithm, shown in Fig. 4.4, shifts the evaluation window for each pixel to process the whole image. The center

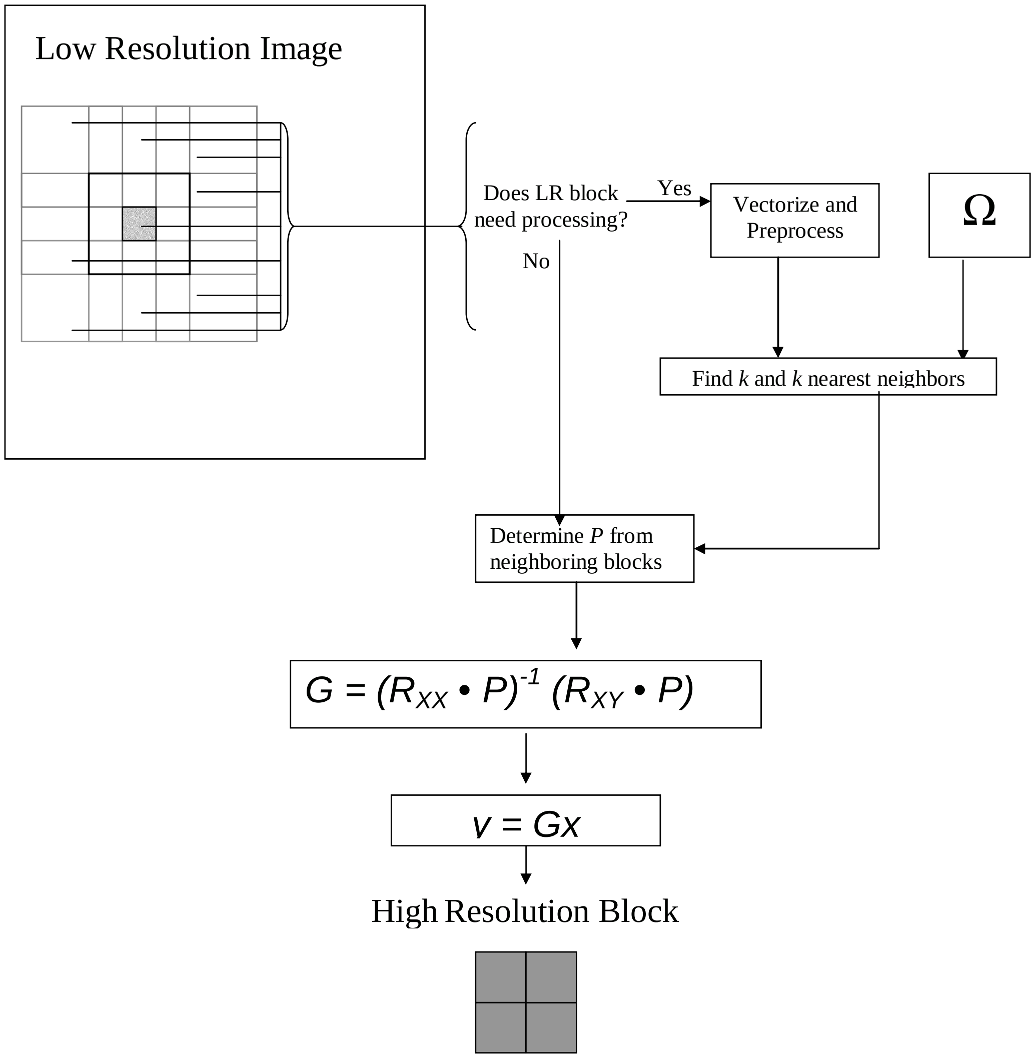

Low Resolution Image
Does LR block need processing?
Yes
No
Vectorize and Preprocess
Ω
Find k and k nearest neighbors
Determine P from neighboring blocks
G = (R_XX • P)^-1 (R_XY • P)
y = Gx
High Resolution Block

Figure 4.4: Description of Windows and Blocks in Adaptable k-NN

block in Fig. 4.3 depicts the block that is to be interpolated. The surrounding blocks supply ψ values to weight the consideration of individual neighbors. All low-resolution blocks have or need to calculate k, and the entire algorithm Fig. 4.4 uses these blocks in the evaluation window to determine G and the high-resolution patch. In raster scans, for the majority of the time, the only low-resolution block for which neighbors have not been calculated is the lower right block.

4.5 Complexity Issues and Specifics on Implementation

Nearest neighbor algorithms are notorious for their runtime complexity. Recently, there has been a fair amount of incentive for search speeds of k-NN algorithms to improve, which has lead to an extensive and diverse collection of literature in the area. This section reviews (albeit superficially) contributions that aid k-NN searches by remaining efficient as N grows large and ultimately choosing a single method tailored to the problem at hand.

There are several directions for optimization and speedups in k-NN searches: structured searches, approximation by lowered dimensionality (partial distances), and the assignment/editing of prototypes in the training set, all of which are described in [50] and [105]. Structured searches [21, 42, 64, 89] transfer the complexity at runtime to training through domain partitioning, using structures such as trees or vector quantization. Partial distance searches [105, 60, 59] that are locally determined calculate distances with a subset of dimensions in $\mathbb{F}$ indicating whether to pursue certain training points. Finally, training set editing [109] deletes points that may seem irrelevant, are duplicates, or those that are likely to be erroneous or cause errors. By the same token, if possible, editing could conceivably enhance points that are important and appear frequently. In reality, a combination of the algorithmic techniques define any significant reduction in computational burden, where it was concluded in [60] that exact neighbor searches cannot yield reasonable complexity.

Overall surveys of bodies of work involving k-NN search-time reduction [59, 109, 14] have proven helpful, and elements from several techniques have been incorporated into our algorithm. For example, 5×5 low resolution blocks equal 25 dimensions. We improve efficiency by 30% when avoiding the remainder of distance calculations if the distance is known accurately enough. Finally, we can reduce number of points in the training set, effectively lowering N, by manually extracting duplicate or erroneous data points. All these techniques do not significantly alter the rate of error performance (sometimes even boosting visual quality) but our primary speed concerns remain unresolved, which are addressable in the number of distance evaluations.

An improvement in complexity to the degree we are seeking necessarily dictates some loss in precision, meaning the employment of several of the more heavily approximated, inexact algorithms. A good example of speedy k-NN search structure are tree approximation algorithms such as [89]. Both [49] and [9] have realized their algorithms through a version of tree-based methods. The tree structures often use directional splitting according to the largest principle component, the eigenvector with the largest eigenvalue. Instead of calculating the costly matrix operations that go along with this method, inspired by previous work in kernel learning [84] and evidence to corroborate that intrinsic dimensionality is often very low, an especially effective flavor of approximation introduces the concept of local sensitivity hashing (LSH) [59], where runtime search approximations are drawn from *near* neighbors [105] as opposed to the exact *nearest* neighbor. Closely akin to LSH algorithms and the aforementioned tree algorithms are random projection (RP) trees. A detailed description in [37] is interested in automatically adapting to intrinsic low-dimensional structures through RP trees without explicitly learning this structure. Several authorities in the field have boasted that RP trees can perform in real time (using programmable gate arrays) with training data quantities on the order of tens of millions with dimensionality at 10×10.

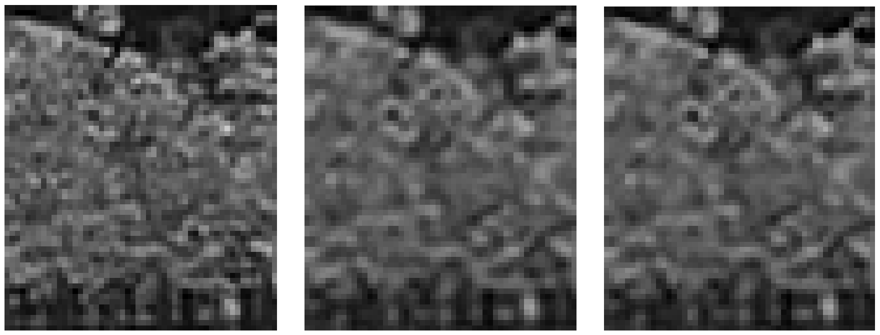
The analysis of all of these algorithms are beyond the scope of this investigation, but their relevance to the problem at hand mandates a wise choice that is reflective of the functional domain and problem at hand. With the sheer quantity of image data, a choice of structured tree searches sufficiently covers the necessary requirements without sacrificing too much accuracy. Combined with several of the calculation-evading tactics, the runtime of search-trees should be more than acceptable.

4.6 Results

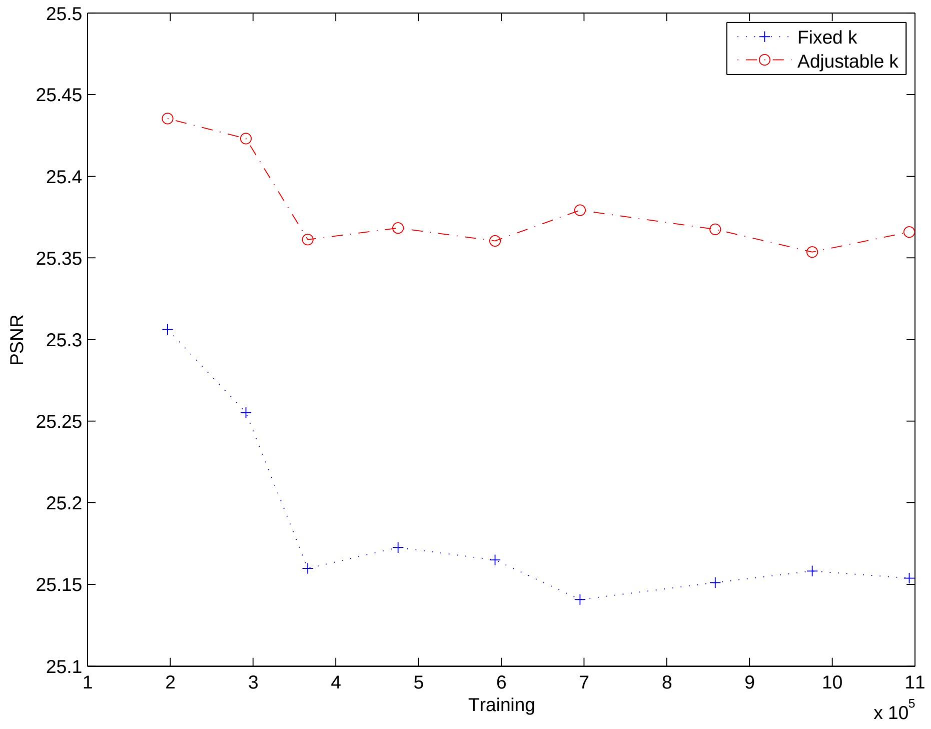
Given the occurrences of insufficient data in Fig. 3.3, the assumption that a very large training set is at the proposed algorithm's disposal remains (common for any k-NN algorithm). The scale is highlighted with the proposed algorithm's failure to yield meaningful results when the high-resolution test image is available as the only training. Hence, our experiments mandate a minimum of 12 to 14 large images, where resources consist of millions of image patches. The phenomenon returns to previous explanations of determining filter coefficients and the proposed algorithm's choice of k. As it turns out, creating a nonsingular matrix is surprisingly difficult, and very often insufficient data plagues the construction effort.

Images are diverse entities, but for brevity and analysis purposes, they are divided into two categories in Fig. 4.6 and Fig. 4.5: edges and texture. Both figures compare the performance of a variable k versus a fixed k in nearest neighbor weighted filtering. Fig. 4.5 compares adaptable k-NN to fixed k k-NN of texture. The proposed algorithm is consistent in superior qualitative performance for any texture that is tested. Fig. 4.6 compares adaptable k-NN to fixed k k-NN of edges. The proposed algorithm sacrifices some numerical performance to ensure visually pleasing results. Adjustable k PSNR results in Fig. 4.6(d) are somewhat lower than fixed k results. Given the behavior of k-NN in edge image patches, seen in Fig. 3.3, the performance is justified by some intuition.

For most edge content, the amount of training that is "close" to $\mathbf{x}_t$ is insufficient, which is manifested in some ambient noise in Fig. 4.6(c). The non-adjustable k cannot pool from additional data though the neighbors it uses are not very "relevant". The adjustable k-NN algorithm will, on the other hand, attempt to grab more data to obtain a general result, and the image becomes softer with fewer errors in Fig. 4.6(b). Nevertheless, in obtaining a very general filter, the adjustable k-NN sacrifices high PSNR values because it uses a very general and broad filter that averages more than it specializes. Hence, the fixed k-NN will have higher PSNR values.



(a) Original Image (b) Adjustable k (c) Non-adjustable k

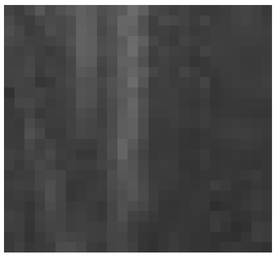


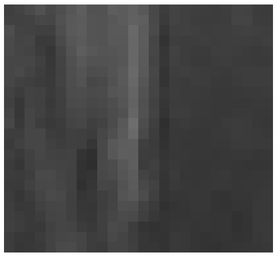
(d) PSNR - Texture

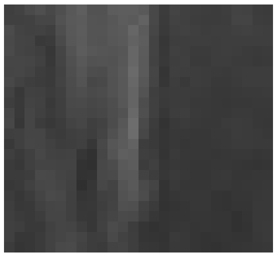
Figure 4.5: Texture Comparisons of an Adaptable k vs. Fixed k

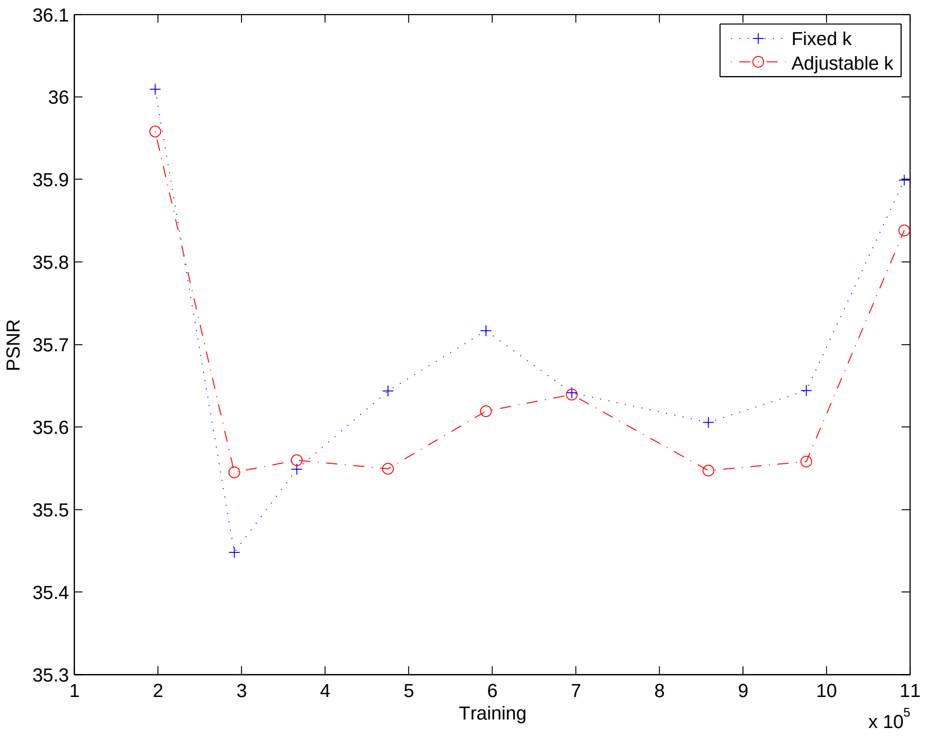
On the opposite spectrum, texture, according to Fig. 3.3, has a strong showing in most training sets. Therefore, the quantitative values in Fig. 4.5(d) reflect specific filters that are specially designed for the situation determining a quantitatively superior result.

From descriptions in Sec. 4.2, values of k are directly correlated with the


(a) Original Image


(b) Adjustable k


(c) Non-adjustable k



(d) PSNR

Figure 4.6: Edge Comparisons of an Adaptable k vs. Fixed k

amount of training, N. Increasing N usually results in decreasing k, and this is verified through experimentation because k tended to stay around select values, k_{avg}, for particular N. In Fig. 4.8 and Fig. 4.9, k had a standard deviation of 59.79 on average staying around 151.76. Often $k_{max} = 10^6$ was reached, i.e. $k^* \geq \zeta$, but it was usually either hit or miss, where either k would be close to k_{avg} or ζ was reached, in which case, image values were created using bicubic interpolation.

We can alter two degrees of freedom in our experiments, σ, the bandwidth parameter of the Gaussian used to extend around the test point, or η, the minimum number of training points necessary. Between the two, it is easier to alter η because bandwidth σ is a squared exponent term and is difficult to control.

Fig. 4.7 depicts the role of η, where the images are placed in logarithmically increasing order of η. While Fig. 4.7(a) appears clearer and sharper, there appears to be quite a number of errors that afflict the construction effort. Meanwhile, Fig. 4.7(c), appears blurrier and more washed out, but there are fewer errors.

4.6.1 Comparisons to Non-Learning-Based Algorithmic Results

The proposed algorithm naturally appears superior over state of the art interpolation algorithms in Fig. 4.8. The comparisons are probably not very fair, because the amount of information available to the proposed algorithm is greater than any of those compared to it. Numerical results in Table 4.1 are also poor.

4.6.2 Comparison to Learning-Based Algorithmic Results

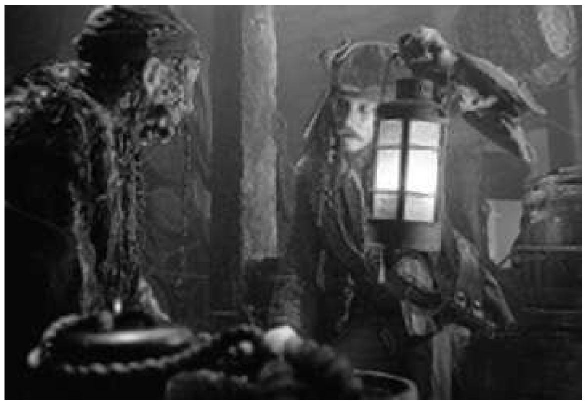
Fig. 4.9 shows the qualitative results of other nearest neighbor and statistical classification algorithms alongside our own, and Table 4.1 gives the quantitative results[5] for various images. The total number of training for this set of test runs was $N = 4,309,914$ points.

Fig. 4.9(b) and Fig. 4.10(b) show a decidedly edge-centric result, which can be explained by the neighborhood regularization done through Markov networks. The algorithm does especially well in areas where the original image contains a disparity between textures such as the border between the bus and the background,

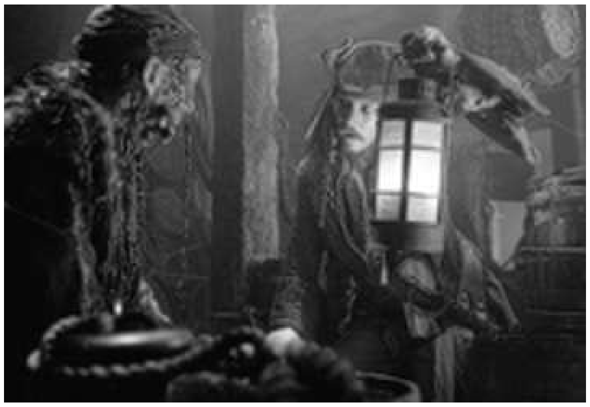
[5]Peak Signal-to-Noise Ratio (PSNR) is the widely accepted and commonly used standard of quantitatively measuring image quality.

$$\text{PSNR} = 10 \cdot log_{10} \frac{255^2}{MSE}.$$

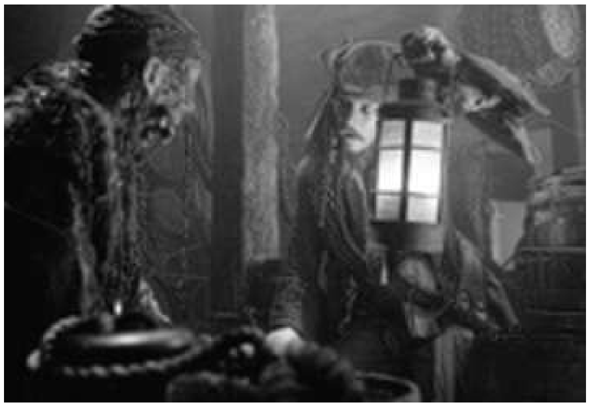
The value 255 is used because it is the maximum image value. MSE stands for mean squared error, the standard definition.



(a) Minimum k, $\eta = 50$

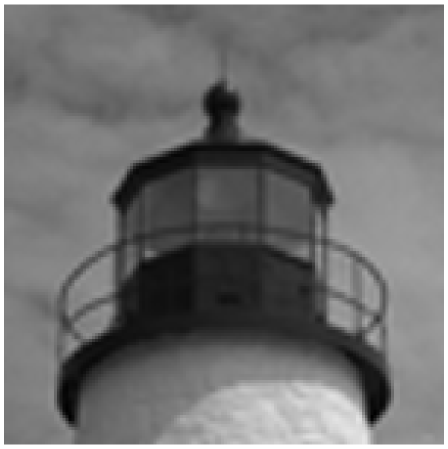


(b) Minimum k, $\eta = 500$

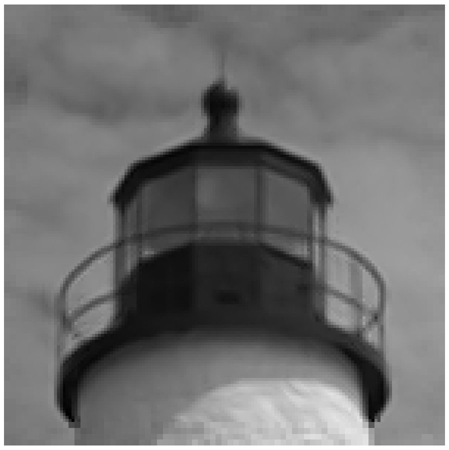


(c) Minimum k, $\eta = 5,000$

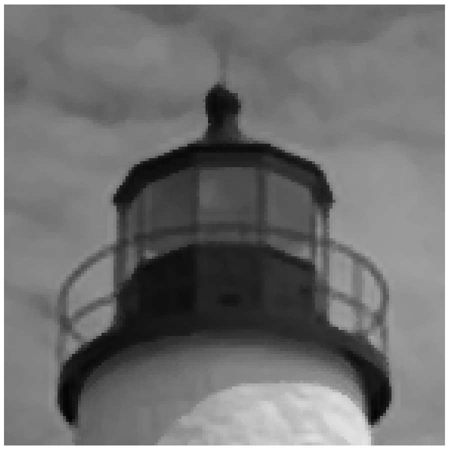
Figure 4.7: Effect of Varying $\eta = k_{min}$



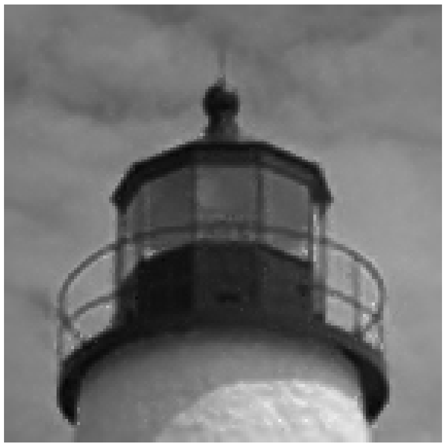
(a) Bicubic Interpolation



(b) Edge Directed Interpolation [74]



(c) Subpixel Edge Localization [62]



(d) Adaptive k-NN

Figure 4.8: Comparisons to State of the Art Interpolation Techniques

but gives average performance where edges are less well-defined (see the pole above the bus in Fig. 4.9(b)). The same phenomenon can be seen for involved textures (see the barrel and pillar in Fig. 4.10). This result is interesting, and could be a byproduct of the particularly large emphasis that Freeman places on texture regularity. More likely, Sec. 4.3 reasons that the trend could be due to an incorrect choice of a single neighbor from insufficient edge information. Fig. 4.9(e) and Fig. 4.10(d) follow this model, but here, the benefits of k rather than a single neighbor becomes apparent, significantly reducing the same artifacts that afflict [49].

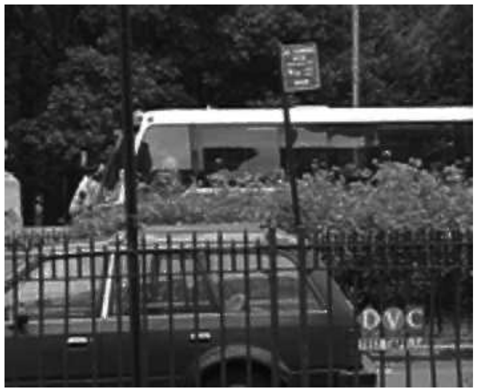
Table 4.1: Adaptable k-NN PSNR comparisons

METHOD	**PSNR Values**		
	Pirates	Lighthouse	Bus,4
Bicubic	29.28 dB	28.87 dB	24.55 dB
NEDI [74]	26.82 dB	27.44 dB	22.57 dB
SEL [62]	26.67 dB	27.38 dB	22.63 dB
128 Class RS [9]	28.91 dB	28.98 dB	26.48 dB
LLE [28]	21.97 dB	22.70 dB	18.04 dB
Example-Based [49]	27.28 dB	28.75 dB	25.52 dB
Proposed Algorithm	29.62 dB	29.12 dB	26.25 dB

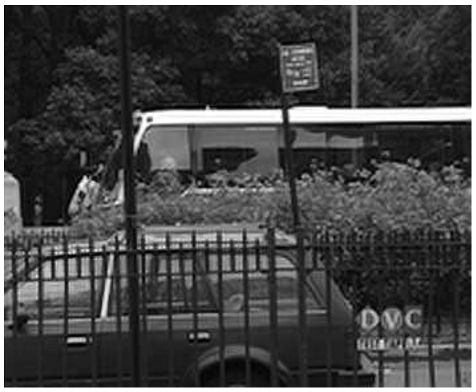
Again, our algorithm does not have the luxury of an initial interpolation stage, instead filtering from scratch, and when there is insufficient training, cross-validation for η and σ is difficult albeit possible. Owing to this fact, the proposed algorithm manages smaller N sizes at the lower end of $\alpha < 10$ in $\alpha \times 10^5$ worse than [49]. Yet, when N is large, it often outperforms all statistical methods, the conclusion here being that the proposed algorithm represents the inherent relationship well.

Fig. 4.10 compares the performance of various methods of interpolation in the “Pirate” images. [9] seems to be oversharp in the areas such as the face, the side of the hat. Meanwhile [49] concentrates on edge-continuity fairly well, but

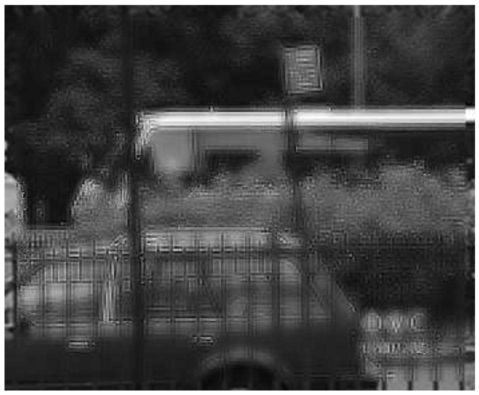
(a) Original Image



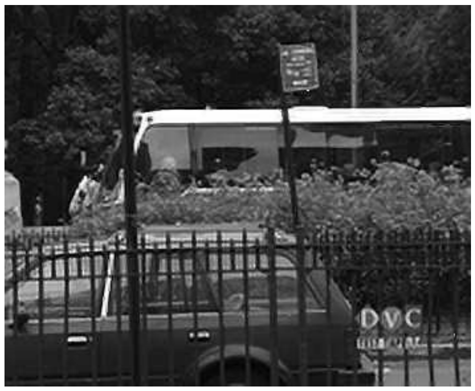
(b) Example-based Superresolution [49]



(c) Classification-based Interpolation [9]



(d) Neighbor-Embedding [28]



(e) Adaptive k-NN

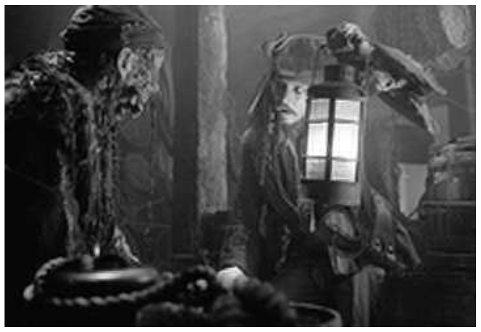
Figure 4.9: Comparisons to Statistical Learning Methods, Bus Images

adaptive k-NN elucidates the texture of the image. (Notice the barrel, the wooden pillar in the background, hat, etc.)

Interestingly enough, while [28] is the most similar in theory to our algorithm, the results (Fig. 4.9(d) and in general) did not reflect this. No experiment adjusting any parameter would yield acceptable results. Experiments in [28]

(a) Bicubic Interpolation

(b) Example-based [49]


(c) Classification-based Interpolation [9]

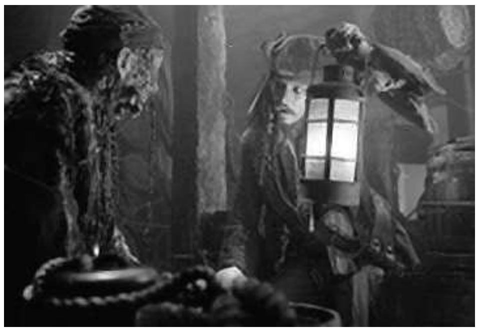

(d) Adaptive k-NN

Figure 4.10: Comparisons to Statistical Learning Methods, Pirate Images

trained on single images, but the feature space of [28] in those cases was an astonishing 100 dimensions, for which we would usually expect large N. Surprisingly, even with large N, the algorithm still remains at a disadvantage; our implementation[6] of [28] was never able to achieve the generalization of any of the other statistical learning algorithms, working for a select few images. In fact, additional postprocessing was required to rid the constructed image of speckle noise from erroneous estimation. The errors may be due in part to the fact that neighboring patch information is not considered. Because the neighborhood preservation rate of the output patch is on average less than 10% [112], we can expect little continuity in the image result. Another possible explanation attributes the errors to small k values, usually less than 40, which prevents the contribution of large amounts of

[6]This was later independently verified by the author of the original work, [112], and code from the author of [28] through e-mail correspondences.

information to offset estimation errors, i.e. k often did not scale proportionately with N. The proposed algorithm with $N \approx \alpha \times 10^6$ found k on average in the high hundreds, which oddly enough, applied to [28] gave even poorer visual quality.

Fig. 4.9(c) and Fig. 4.10(c) from [9] give slightly clearer results than the other statistical methods, but introduce sharpening errors in many instances. This is due in part to the number of classes involved (128) because the performance varies with different class sizes. Intuitively, the two methods should converge when the k in k-NN algorithms is excessively large and the number of classes in classification-based algorithms[7] is large as well. This is eventually true, but the way in which this happens is unexpectedly non-monotonic. Ultimately, the tradeoff pits the number of data points per class against the number of classes, where N plays an obvious role. Current results show that as N increases (as well as the number of classes), the role of k-NN plays less, and classification based algorithms inexplicably do better. Barring this broad analysis, which composes the subject of current work, the only relevant conjecture made here is that conventional classification-based algorithms could stand to improve due their deficiency of classes.

Additional comparisons and results can be found at `http://videoprocessing.ucsd.edu/~karl/k-NN/`.

4.7 Summary

A k-NN algorithm with optimal filters and a variable k, determined by relevant training, has been proposed, tested, and compared to the state of the art. The analysis of this algorithm leads to several conclusions:

1. For small training sets, edges cannot be accurately depicted with any nearest neighbor algorithm (if the metric used is Euclidean distance).

2. For large training sets, k-NN performs especially well, both quality and quantity-wise.

[7][9] grows a tree for training, but then prunes the tree. Comparisons for our purposes deal with overfitting by throwing more data at the problem, rationalizing the omission of the pruning process to the sheer quantity of training.

3. Fast neighboring-patch approximations of a Markov Network elucidate edges and provide good continuity but sometimes hinder texture synthesis in cases where the training is limited.

4. Linear filtering is a good mask and covers up considerable estimation errors.

5. The direct application of k-NN regression with slight modifications exhibits competitive image quality and offers detailed texture.

The result of the enumerated items is an interpolative technique that pools together concepts from several existing works to create higher resolution image content from neighborhood information given in a training set.

4.8 Acknowledgements

The text of Chapter 4 is adapted from *Adaptive k-Nearest Neighbor*, Karl Ni and Truong Nguyen, January 2008, manuscript submitted to *IEEE Transactions on Image Processing*. Portions of the chapter also appear in "An Adaptive k-Nearest Neighbor Algorithm for Image Interpolation", Karl Ni and Truong Nguyen, in Proceedings of the *IEEE International Conference on Image Processing*, April 2008. The dissertation author was the primary author of these publications, and the co-author listed directed and supervised the research which forms the basis for this chapter.

5 Support Vector Regression Image Interpolation

A poor idea well written is more likely to be accepted than a good idea poorly written

—Isaac Asimov

Up until now, the function estimation in previous chapters as well as several referenced algorithms [65, 9, 49, 28, 24] approximate relationships using linear regression. As image content is exceedingly diverse, assumptions on linearity might prove too restrictive. In other words, MMSE interpolative and linear regression models may not be sufficiently representative of the true relationship between low and high-resolution information. Chapters 5 and 6 explore nonlinear regression techniques as an alternative in an effort to increase estimation performance.

Known mathematical regression models come in several forms: linear, Bayesian, polynomial, etc.. As it turns out, the static function models just mentioned are severely limited in their domain space, and may be too restrictive to represented image content well. Instead, it is more effective to use functionals rather than a single function in a high dimensional feature space to learn the relationship. This can be realized with support vector machines applied to regression (SVR).

SVR simultaneously provides function estimation while offering nonlinear complexity by using a high-dimensional mapping, $\phi(\mathbf{x})$. This chapter investigates the potential of SVR in the image superresolution problem by placing it in various frameworks and adding structural information that might aid in solving for missing high-resolution information. The frameworks discussed in this chapter include

1. the general application of SVR to superresolution in a localized manner,
2. optimal choices of the high-dimensional mappings in SVR by adjusting the kernel function,
3. an alternate look at the domain of the training set based on inherent structural advantages.

The enumerated frameworks are discussed throughout the chapter, which is organized as follows. Sec. 5.1 introduces, reviews, and explains SVR to give the necessary background of its application to the superresolution problem. Then, Sec. 5.2 explores a method of learning the kernel matrix, an essential element of SVR. Next, the framework of the image superresolution problem through a general SVR approach is established in Sec. 5.3. Finally, Sec. 5.4 improves the solution by observing a change in domain.

5.1 Support Vector Regression

The support vector machine (SVM), originally proposed in [124], is a learning algorithm [35][92] with the ability to provide function estimation. By using a mapping, $\Phi : \mathcal{X} \rightarrow \mathcal{F}$, where $\mathcal{X}$ is the domain and $\mathcal{F}$ is usually a high-dimensional feature space, support vector regression (SVR) operates in feature space to approximate unknown functions in an output space $\mathcal{Y}$, thereby using nonlinear functions to linearly estimate an unknown regression.

Suppose that we are given a training set with N input-output pairs as in (5.1).

$$\Omega \ = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\} \tag{5.1}$$

where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. Then, we can estimate the function $f : \mathbf{x} \rightarrow y$ by utilizing the feature space through ϕ by the following optimization. In this manner, SVR models a way to generalize unseen inputs to what has been observed in the training set as the following optimization problem.

$$\begin{aligned}
\min_{\mathbf{w},b,\xi^{+/-}} \quad & \left(\tfrac{1}{2}||\mathbf{w}|| + C\sum_{i=1}^{N}\left(\xi_i^{+} + \xi_i^{-}\right)\right) \\
\text{s.t.} \quad & (\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b) - y_i \leq \varepsilon + \xi_i^{+} \\
& y_i - (\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b) \leq \varepsilon + \xi_i^{-} \\
\text{and} \quad & \\
& \xi_i^{-}, \xi_i^{+} \geq 0, \qquad (5.2)
\end{aligned}$$

for all $i \in [1, N]$ and where $\mathbf{w} \in \mathcal{F}$ [103].

Notice that for $y \in \mathcal{Y}$, $\forall\ i \in [1, N]$, there are two inequalities that bound the output training: one for the upper boundary and one for the lower boundary. Meanwhile, slack variable vectors ξ_i^{+} and ξ_i^{-} correspond to the upper and lower parameters in which the function $g(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x})\rangle + b$ is allowed to deviate for a prespecified error and cost, $[\varepsilon, C] \geq \mathbf{0}^T$. We can write the Lagrangian and express the dual maximization problem (5.3).

$$\begin{aligned}
\max_{\alpha^{+},\alpha^{-}} \quad & -\tfrac{1}{2}\sum_{i,k}\left\{(\alpha_i^{+} - \alpha_i^{-})(\alpha_k^{+} - \alpha_k^{-})K(\mathbf{x}_i, \mathbf{x}_j)\right\} \\
& -\epsilon\sum_i(\alpha_i^{+} + \alpha_i^{-}) + \sum_i y_i(\alpha_i^{+} - \alpha_i^{-}) \\
\text{s.t.} \quad & \sum_i(\alpha_i^{+} - \alpha_i^{-}) = 0 \\
& 0 \leq \alpha_i^{+/-} \leq C \qquad (5.3)
\end{aligned}$$

with the regression estimate as

$$g(\mathbf{x}) = \sum_i(\alpha_i^{+} - \alpha_i^{-})K(\mathbf{x}, \mathbf{x}_i) + b \qquad (5.4)$$

where a dot product in $\mathcal{F}$ is defined by $K(\mathbf{s}, \mathbf{t}) = \langle \phi(\mathbf{s}), \phi(\mathbf{t})\rangle$, the kernel function.

5.2 Kernel Learning for Support Vector Regression

Using a mapping $\Phi : \mathcal{X} \mapsto \mathcal{F}$, SVR is often better suited to represent complicated relationships that we otherwise could not realize linearly. Within $\mathcal{F}$, a kernel function written as a kernel matrix is defined to be a collection of dot products for an arbitrary Φ that may or may not be known. Using the kernel matrix K, computational complexity is reduced because the actual high-dimensional mapping in determining $d = \langle \phi(\mathbf{s}), \phi(\mathbf{t}) \rangle$, which is quite often intractable, is unnecessary when solving (5.3). This definition also allows Φ to be unknown, in which case, K can be conceptually chosen to be a desired similarity metric depicting the "nearness" of two vectors. Thus, the selection of the kernel matrix K becomes important and should be sensitive to the training data.

Ordinarily, a single kernel matrix, usually a factor decided by human decision, is selected from a set of precalculated kernels to some degree of accuracy if sufficiently cross-validated. Rather than doing this, several works have explored the prospect of learning the kernel matrix [70], [97], [84], [131],[68],[13]. Of particular interest is [70] in which a linear combination of known kernels is optimized to produce a large kernel with good feature representation for the classification problem. The motivation behind this section is that the impossible task of cross-validating all possible combinations of precalculated kernels to determine an optimal one can be theoretically instead of analytically derived.

Within the ideas proposed in [70] is the possibility of incorporating multiple data sources to describe inherent vector space relationships by using multiple kernels. We can choose which features to use (i.e. dot products of $\mathbf{x}' = [x_1, x_3, \ldots]^T$ or $\mathbf{x}' = [x_2, x_5, \ldots]^T$, etc.) and how they will used (i.e. RBF kernels, polynomial kernels, etc.) The extra degrees of freedom fit especially well with our design because the vector space for our particular problem is multi-dimensional compounded by the seemingly desultory nature of local image content. As will become clearer later in the section, it is often the case in high dimensional spaces that individual dimensions of the input vector $\mathbf{x}$ relate differently in the actual feature space. That is to say, individual features may be more relevant than others (feature selection or weighting) or contribute differently (Hilbert space selection).

Like the classification case in [70][1], the analogous optimization for regres-

[1]Lanckriet et. al.[70] offers a method to both inductively and transductively learning a kernel matrix, but the domain in superresolution is too large to predict input vectors beforehand and therefore induction is exclusively used.

sion has been explored in [97], although errors lead the derivation to an incorrect outcome. The errors have been addressed in a previous work [84] and will be reiterated here, supplying a reformulation of a semi-definite programming (SDP) and quadratically constrained quadratic programming (QCQP) problem to learn the kernel for regression in much the same way that it has been derived for classification.

5.2.1 The SDP Problem

Autonomously learning a kernel matrix is generally formulated as a convex optimization problem, where a single optimum (both local and global) avoids initialization problems. A branch of convex optimization dealing with the convex cone of positive semidefinite matrices and its convex subsets can be solved with programming techniques in SDP. This relates to the kernel matrix learning problem because the range of any convex function of kernel matrices, which by definition are positive definite matrices, is convex and lies in the cone of positive definite matrices and hence can be formulated in terms of an SDP problem.

Implicit within this statement is the desire to include data from several sources by using not one single kernel matrix but a convex function of several kernel matrices while keeping the redundancy as low as possible. We do this by choosing from a subset of $S = \{K_i\}$ such that the final kernel matrix K is some convex function of $S_c \subseteq S$. Later in the section, for purposes of developing an SDP, the convex function is chosen to be a linear combination of the K_i matrices in S_c.

An SDP problem is defined to be a convex optimization problem of the form

$$\begin{array}{ll} \min\limits_{\mathbf{u}} & \mathbf{c}^T\mathbf{u} \\ \text{s.t.} & F^{(j)}(\mathbf{u}) = F_0^{(j)} + u_1 F_1^{(j)} + \ldots + u_q F_q^{(j)} \succeq 0 \\ & A\mathbf{u} = b \end{array} \quad , \tag{5.5}$$

for a specified number of j and where $F_i^{(j)}$ are square matrices and $\mathbf{u} \in \mathbb{R}^q$.

The expression for kernel optimization is the SVR equation, and we start from the dual optimization problem in (5.3). For simplification, let $\mathbf{e}$ be a vector of all ones, $\mathbf{y} \in \mathbb{R}^D$ be a vector of labels $y_1, y_2, \ldots, y_D$, and

$$\begin{aligned} \boldsymbol{\alpha}^+ + \boldsymbol{\alpha}^- &= \boldsymbol{\beta}^+ \\ \boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^- &= \boldsymbol{\beta}^- \end{aligned} \tag{5.6}$$

Substitutions in (5.6) eventually lead to the inverting of the kernel matrix K, which is entirely possible for correctly chosen K_i. This is also where our work diverges from [97] because their substitutions involve inverting a matrix defined as $Q(K) = \begin{pmatrix} K & -K \\ -K & K \end{pmatrix}$, an impossible task because $Q(K)$ can at most be rank $\frac{dim(Q)}{2}$ and is thus non-invertible. Invertibility in our derivations is a property that is essential for the use of the Schur complement lemma, which will be discussed shortly.

(5.3) can be rewritten in terms of the substitutions in (5.6) as the optimization problem in (5.7).

$$\begin{aligned} \max_{\boldsymbol{\beta}^+, \boldsymbol{\beta}^-} \quad & -\tfrac{1}{2}\boldsymbol{\beta}^{-T} K \boldsymbol{\beta}^- + \mathbf{y}^T\boldsymbol{\beta}^- - \epsilon e^T \boldsymbol{\beta}^+ \\ \text{s.t.} \quad & \mathbf{e}^T\boldsymbol{\beta}^- = 0 \\ & 0 \preceq \boldsymbol{\beta}^+ + \boldsymbol{\beta}^- \preceq 2C \\ & 0 \preceq \boldsymbol{\beta}^+ - \boldsymbol{\beta}^- \preceq 2C \end{aligned} \tag{5.7}$$

The optimal objective value of (5.7) is the point-wise supremum of affine functions in K, so it is a convex function of K [19]. Thus, it can be optimized with respect to K to yield an optimal kernel matrix. To ensure good generalization, the trace of K is constrained [70], and the corresponding optimization problem can be expressed in terms of its Lagrangian function as

$$\min_K \max_{\boldsymbol{\beta}^{+/-}} \min_{\lambda, \mathbf{q}_{l,u}^{+/-}} \mathcal{L}(K, \boldsymbol{\beta}^{+/-}, \lambda, \mathbf{q}_l^{+/-}, \mathbf{q}_u^{+/-}) \tag{5.8}$$

where we minimize with respect to K and have introduced the Lagrangian variables λ, $\mathbf{q}_{u/l}^{+/-} \succeq 0$, and constrained our solution with $trace(K) = c$. The Lagrangian $\mathcal{L}$

in (5.8) is written in (5.9).

$$\begin{aligned}
\mathcal{L}(K,&\boldsymbol{\beta}^{+/-},\lambda,\mathbf{q}_l^{+/-},\mathbf{q}_u^{+/-}) = \\
&-\frac{1}{2}\boldsymbol{\beta}^{-T}K\boldsymbol{\beta}^{-}+\mathbf{y}^T\boldsymbol{\beta}^{-}-\epsilon\mathbf{e}^T\boldsymbol{\beta}^{+}+\lambda e^T\boldsymbol{\beta}^{-} \\
&\quad+(\boldsymbol{\beta}^{+}+\boldsymbol{\beta}^{-})^T\mathbf{q}_l^{+}+(\boldsymbol{\beta}^{+}-\boldsymbol{\beta}^{-})^T\mathbf{q}_l^{-} \\
&\quad-(\boldsymbol{\beta}^{+}+\boldsymbol{\beta}^{-}-2C\mathbf{e})^T\mathbf{q}_u^{+} \\
&\quad-(\boldsymbol{\beta}^{+}-\boldsymbol{\beta}^{-}-2C\mathbf{e})^T\mathbf{q}_u^{-}
\end{aligned} \tag{5.9}$$

(5.7) is a convex optimization problem and the constraints are strictly feasible. Therefore, from Slater's conditions [118], strong duality holds and we can exchange the order of the maximum and minimum.

$$\begin{aligned}
\min_{K,\lambda,\mathbf{q}_{l,u}^{+/-}}\ \max_{\boldsymbol{\beta}^{+/-}}\quad & \mathcal{L}(K,\boldsymbol{\beta}^{+/-},\lambda,\mathbf{q}_l^{+/-},\mathbf{q}_u^{+/-}) \\
\text{s.t.}\quad & \mathbf{q}_l^{+},\mathbf{q}_l^{-},\mathbf{q}_h^{+},\mathbf{q}_h^{-}\succeq 0 \\
& trace(K)=c \\
& K\succeq 0
\end{aligned} \tag{5.10}$$

In terms of $\boldsymbol{\beta}^{+}$ and $\boldsymbol{\beta}^{-}$, (5.10) is an unconstrained quadratic optimization problem and hence can be analytically solved. After writing the Lagrangian, we set the derivative with respect to $\boldsymbol{\beta}^{+}$ and $\boldsymbol{\beta}^{-}$ equal to zero to obtain

$$\begin{aligned}
\left(\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}^{+}}\triangleq 0\right) &\Rightarrow \quad -\epsilon\mathbf{e}+\mathbf{q}_l^{+}+\mathbf{q}_l^{-}-\mathbf{q}_u^{+}-\mathbf{q}_u^{-}=0 \\
\left(\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}^{-}}\triangleq 0\right) &\Rightarrow \boldsymbol{\beta}_{opt}^{-}=K^{-1}(\mathbf{y}+\lambda\mathbf{e}+\mathbf{q}_l^{+}-\mathbf{q}_l^{-}-\mathbf{q}_u^{+}+\mathbf{q}_u^{-})
\end{aligned} \tag{5.11}$$

Aside from both $\boldsymbol{\beta}$'s, we can eliminate an additional variable, so we substitute for $\mathbf{q}_l^{+}$ such that

$$\begin{gathered}
\mathbf{q}_l^{+}=\epsilon\mathbf{e}+\mathbf{q}_u^{+}+\mathbf{q}_u^{-}-\mathbf{q}_l^{-} \\
\Downarrow \\
\boldsymbol{\beta}^{-*}=K^{-1}(\mathbf{y}+\lambda\mathbf{e}+\epsilon\mathbf{e}+2\mathbf{q}_u^{-}-2\mathbf{q}_l^{-})
\end{gathered} \tag{5.12}$$

Let

$$\boldsymbol{\gamma}=(\mathbf{y}+\lambda\mathbf{e}+\epsilon\mathbf{e}+2\mathbf{q}_u^{-}-2\mathbf{q}_l^{-}). \tag{5.13}$$

Then, we rewrite the objective function of (5.10) using (5.11) to obtain the expression:

$$\frac{1}{2}\boldsymbol{\gamma}^T K^{-1}\boldsymbol{\gamma} + 2C\mathbf{e}^T(\mathbf{q}_u^+ + \mathbf{q}_u^-)$$

Therefore, the resulting optimization problem is

$$\begin{aligned} \min_{K,t,\lambda,\mathbf{q}_u^+,\mathbf{q}_l^-,\mathbf{q}_u^-} \quad & t & (5.14)\\ \text{s.t.} \quad & t \geq \tfrac{1}{2}\boldsymbol{\gamma}^T K^{-1}\boldsymbol{\gamma} + 2C\mathbf{e}^T(\mathbf{q}_u^+ + \mathbf{q}_u^-) & \\ & \epsilon\mathbf{e} + \mathbf{q}_u^+ + \mathbf{q}_u^- - \mathbf{q}_l^- \succeq 0 & \\ & \mathbf{q}_u^+, \mathbf{q}_l^-, \mathbf{q}_u^- \succeq 0 & \\ & K \succeq 0 & \\ & trace(K) = c & (5.15) \end{aligned}$$

In (5.14), the variable K^{-1} in the first constraint brings up an important technique in the formulation of many SDP problems: the Schur complement lemma. The Schur complement lemma is useful in that it allows constraints to be expressed in linear matrix inequality (LMI) form. In terms of its usage with the problem at hand, $K \succeq 0$ implies that

$$t \geq \tfrac{1}{2}\boldsymbol{\gamma}^T K^{-1}\boldsymbol{\gamma} + 2C\mathbf{e}^T(\mathbf{q}_u^+ + \mathbf{q}_u^-)$$

$$\Updownarrow$$

$$\begin{pmatrix} 2K & \boldsymbol{\gamma} \\ \boldsymbol{\gamma}^T & t - 2C\mathbf{e}^T(\mathbf{q}_u^+ + \mathbf{q}_u^-) \end{pmatrix} \succeq 0 \quad , \qquad (5.16)$$

where in (5.16), the positive semidefiniteness of the encompassing matrix in the bottom expression has been rewritten as an LMI by considering the Schur complement in the top expression.

With this in mind, (5.14) takes the following form

$$\begin{aligned}
\min_{K,t,\lambda,\mathbf{q}_u^+,\mathbf{q}_l^-,\mathbf{q}_u^-} \quad & t \\
\text{s.t.} \quad & \begin{pmatrix} 2K & \boldsymbol{\gamma} \\ \boldsymbol{\gamma}^T & t - 2C\mathbf{e}^T(\mathbf{q}_u^+ + \mathbf{q}_u^-) \end{pmatrix} \succeq 0 \\
& \mathbf{q}_u^+, \mathbf{q}_u^-, \mathbf{q}_l^- \succeq 0 \\
& \epsilon e + \mathbf{q}_u^+ + \mathbf{q}_u^- - \mathbf{q}_l^- \succeq 0 \\
& K \succeq 0 \\
& trace(K) = c
\end{aligned} \tag{5.17}$$

In order to come up with a meaningful solution that involves our input, we adjust (5.17) to be a solvable SDP problem by writing K as a linear combination[2] of fixed kernels in $S = \{K_i\}$. This avoids a trivial solution, and thus, we now optimize with respect to the coefficients μ_i, of the linear combination in

$$K = \sum_i \mu_i K_i(\cdot,\cdot) \ , \tag{5.18}$$

The final SDP problem is stated in (5.19).

$$\begin{aligned}
\min_{\boldsymbol{\mu},t,\lambda,\mathbf{q}_u^+,\mathbf{q}_l^-,\mathbf{q}_u^-} \quad & t \\
\text{s.t.} \quad & \begin{pmatrix} 2\sum_i \mu_i K_i & \boldsymbol{\gamma} \\ \boldsymbol{\gamma}^T & t - 2C\mathbf{e}^T(\mathbf{q}_u^+ + \mathbf{q}_u^-) \end{pmatrix} \succeq 0 \\
& \mathbf{q}_u^+, \mathbf{q}_u^-, \mathbf{q}_l^- \succeq 0 \\
& \epsilon e + \mathbf{q}_u^+ + \mathbf{q}_u^- - \mathbf{q}_l^- \succeq 0 \\
& \sum_i \mu_i K_i \succeq 0 \\
& trace(K) = c
\end{aligned} \tag{5.19}$$

The result in (5.19) is general with possible application to problems other than superresolution.

[2]In fact, any convex set of K would yield a convex optimization problem. Though a single solution would exist, there may be issues with the complexity of the resulting optimization problem as it may not be an SDP anymore.

5.2.2 The QCQP Problem

A QCQP problem is defined to be a convex optimization problem of the form

$$\begin{aligned} min_{\mathbf{u}} \quad & f_0(\mathbf{u}) \\ \text{s.t.} \quad & f_j(\mathbf{u}) \geq 0, \qquad j = 1, \dots n \end{aligned} \tag{5.20}$$

for a specified number of j and where f_j are quadratic functions of the form $f_j(\mathbf{u}) = (A_j\mathbf{u} + \mathbf{b})^T(A_j\mathbf{u} + \mathbf{b})$.

From (5.19), the QCQP for learning K arises from an added constraint, $\mu_i \geq 0$, which causes some loss of generality, though it does ensure positive definiteness when inductively applying the learned kernel. The intuition behind this is simple; a linear combination of kernels where the coefficients of the combination are guaranteed to be positive will always yield a positive-definite matrix and hence a valid kernel. Mathematically, this is

$$\mu_i \geq 0 \Rightarrow \left\{ \sum_i \mu_i K_i \succeq 0 \Leftrightarrow K \succeq 0 \right\} \ . \tag{5.21}$$

On the other hand, the complexity of the kernel is never simplified because the positive eigenvalues of each $(\mu_i K_i)$ will never reduce kernel rank.

The QCQP problem is derived in the same manner as [70], and is given in (5.22).

$$\begin{aligned} \max_{\boldsymbol{\beta}^+, \boldsymbol{\beta}^-, p} \quad & 2\mathbf{y}^T\boldsymbol{\beta}^- - 2\epsilon\mathbf{e}^T\boldsymbol{\beta}^+ - cp \\ \text{s.t.} \quad & p \geq \boldsymbol{\beta}^- K_i \boldsymbol{\beta}^- \\ & \mathbf{e}^T\boldsymbol{\beta}^- = 0 \\ & 0 \preceq \boldsymbol{\beta}^+ + \boldsymbol{\beta}^- \preceq 2C \\ & 0 \preceq \boldsymbol{\beta}^+ - \boldsymbol{\beta}^- \preceq 2C \end{aligned} \tag{5.22}$$

Here, K_i are the smaller positive semi-definite kernels in (5.18) for kernel construction. The μ_i values come out of the dual Lagrangian variables.

A single optimization variable p may seem to suggest that only one dual variable μ_i is necessary, meaning that $\beta^- K_i \beta^- = p$ is likely satisfied for one i. In low dimensional spaces in which there are fewer non-redundant K_i, this may be the case. In higher dimensional, more complicated spaces (including the vector space defined by our superresolution approach), there may be several μ_i's that simultaneously satisfy equality in the constraint $p \geq \beta^- K_i \beta^-$. This has important ramifications that justify both QCQP and SDP problems over single kernel cross-validation because it implies that there are several nonzero μ_i's. Consequently, several K_i matrices are required to fully describe a sufficiently descriptive Hilbert space, which through (5.22), can be theoretically obtained. The high probability that $||\boldsymbol{\mu}||_0$ is strictly greater than unity further validates the theoretical approach over the impossible task of cross-validating over every linear combination of K_i in $S = \{K_i\}$.

A more intuitive explanation of multiple kernel usage in linear combinations of K_i lies in the complicated description of the regression itself. A single kernel describes a single space of some sort. Typically, unknown regressions cannot be described by a single space, while other spaces may be irrelevant to the regression. This statement comes about because the projection of the regression hyperplane onto individual kernel spaces will usually not be illustrative enough, and the projections onto irrelevant kernel spaces are meaningless and add no value. A linear combination has the capacity to include or exclude these spaces and yield a sufficiently descriptive Hilbert space that is best suited to our needs, picking and choosing the dimensions and spaces that are necessary while ignoring those that are not. For example, take K_i's that are determined by one or multiple combinations of individual features. In this set of K_i matrices, the space that a particular kernel describes takes the form of a dimensional axis. Then, to describe a hyperplane, many dimensions are unnecessary while other dimensions, alone, cannot describe the hyperplane. This is the main idea behind rationalizing $K = \sum_i \mu_i K_i$ and why multiple μ_i of $\boldsymbol{\mu}$ are usually nonzero.

5.3 General SVR Superresolution

With support vector regression and its optimal kernel, the most straightforward administration of SVR to superresolution is described in this section. Given low and high-resolution image patches I_{LR} and I_{HR} with sizes $D \times D$ and $U \times U$ respectively, to superresolve the center pixel of I_{LR} by a factor of U, we define vectors

$$\begin{aligned}\mathbf{x} &= \text{vectorize}(I_{LR}) - \text{center pixel}(I_{LR}) \in \mathbb{R}^{D^2 \times 1} \\ \mathbf{y} &= \text{vectorize}(I_{HR}) - \text{center pixel}(I_{LR}) \in \mathbb{R}^{U^2 \times 1} \qquad (5.23)\end{aligned}$$

in a given training set Ω of $\mathbf{x}_i$ feature and $\mathbf{y}_i$ label pairs. The task at hand is superresolution by a factor of U to predict U high-resolution pixels corresponding to the center pixel of the $D \times D$ patch. For $2X$ superresolution, this is shown in Fig. 5.1 for $D = 5$ and $U = 2$.

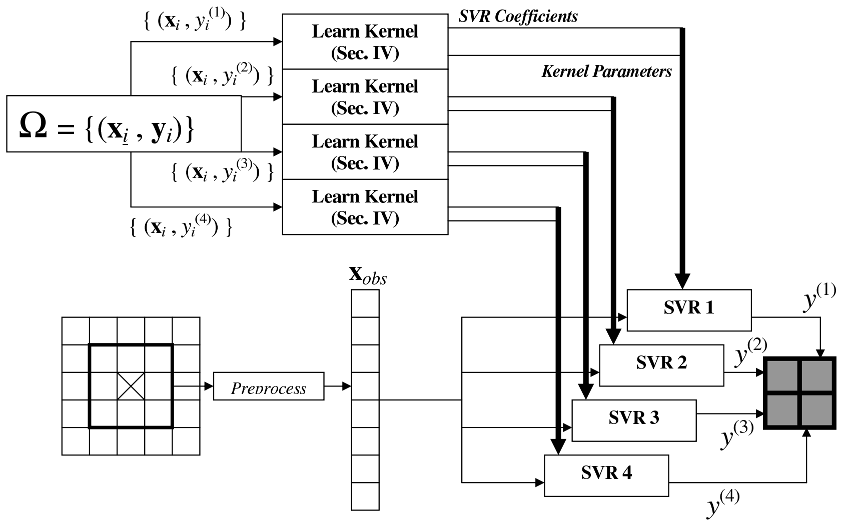


Figure 5.1: Procedure for General SVR Superresolution.

This is a multiple output regression problem and in the literature it is often solved as separate single output regressions. Recently, there has been some work on learning vector valued function in [78][125][113], but we adopt the traditional method of treating multiple output regression problems as separate single output regression problems for each output dimension. Therefore, learning the four outputs becomes $\left\{y^{(j)} = g^{(j)}(\mathbf{x})\right\} \subset \mathbb{R}$ for $j = 1, \ldots, 4$, given the input $\mathbf{x} \in \mathbb{R}^{N^2}$, and $g^{(j)}$ is estimated by SVR in (5.3).

Although the results show that SVR has the capability to provide this regression with fairly clear results, the idea could stand to gain from improvements. A single regressor for a large training set introduces substantial computational complexity. Depending on the data set, the problem quickly become intractable in (5.14) and (5.22), when the kernel matrix size for each $K_i(\cdot, \cdot)$ scales according to N^2 where N is the number of training points. For $K(\cdot, \cdot)$ to be a sum of M small kernels, the required order of memory exceeds $M \cdot N^2$ without even considering other inequality constraints. Also, without further enhancements, this idea relies on the heavy machinery of SVR to recognize all types of image content, which affects the quality of the prediction due to the problem complexity and the large variety of $\mathbf{x}$ in $\mathcal{X}$.

5.4 Discrete Cosine Transform Structure

One way to reduce problem complexity considers inherent domain properties. Structured representations from these properties offer an informative view of the domain and range and assume a simpler relationship. The simpler the relationship, the better the regression.

Training data can define various relationships, the choice of which lends several degrees of freedom. We can take any type, dimension, or combination of inputs, and from this, estimate an equally diverse choice of outputs. A choice where the mapping from input to output is simple alleviates pressure in training and testing, and generally results in a better model. We investigate low-resolution discrete cosine transform (DCT) coefficients as input and considers relevant processes involved in image downsampling to obtain the output. Considering structure involved in downsampling benefits the learning process aiding in interpolation and superresolution. We first observe the one-dimensional case in Sec. 5.4.1 and then apply the same principles to two dimensions.

5.4.1 Decimation in Time

In image upscaling, we are typically trying to recover signal samples that have been removed by decimation. Decimation in time by n retains every n^{th} sample of a signal. For decimation by two, n equals 2, and we retain all even samples while discarding all odd samples.

There are several types of DCTs: DCT-I, DCT-II, etc. Most signal compression algorithms rely on the DCT-II, to compress their signals. The definition of $X_N^{C2}(m)$, the N-point DCT-II, of a signal $x(n)$ is:

$$X_N^{C2}(m) = k_m \sum_{i=0}^{N-1} x(n) cos\left[\frac{(2n+1)m\pi}{2N}\right], \qquad m = 0, 1, ..., N-1. \tag{5.24}$$

As reported by Yip and Rao [99], we can write the N-point DCT as operations on combinations of even and odd samples.

By letting:

$$\begin{aligned} G(m) &= k_m \sum_{n=0}^{\frac{N}{2}} \{x(2n) + x(2n-1)\} cos\left[\frac{mn\pi}{\frac{N}{2}}\right] \\ H(m) &= k_m \sum_{n=0}^{\frac{N}{2}-1} \{x(2n) + x(2n+1)\} cos\left[\frac{(2n+1)m\pi}{N}\right] \end{aligned} \tag{5.25}$$

$$\Rightarrow X_N^{C2}(m) = k_m \left[\frac{G(m) + H(m)}{\cos\frac{mn\pi}{N}}\right], \qquad m = 0, 1, ..., \frac{N}{2} - 1. \tag{5.26}$$

$$\Rightarrow X_N^{C2} = k_m \left\{ \begin{array}{c} DCT\text{-}I(x(2n)) + DCT\text{-}I(x(2n-1)) \\ +DCT\text{-}II(x(2n)) + DCT\text{-}II(x(2n+1)) \end{array} \right\} \tag{5.27}$$

The final (5.27) expression allows us to decompose the image that we wish to be estimated into what is known and what is unknown. Our known signal is $x(2n)$, the even samples, and our unknown signal is $x(2n-1)$ and $x(2n+1)$, the odd samples.

5.4.2 DCT Properties in Two-Dimensions

The amount of information that is lost from decimation in two-dimensions is squared. From (5.27), half of the information comes from the even samples and half of the information from the unknown odd samples. In the 2-dimensional case (decimation in space), only a quarter of information will be known.

Matrix representation can achieve the analogous expressions to those seen in Sec.5.4.1. Let C_1 be the $\frac{N}{2}$-point DCT-I matrix and C_2 be the $\frac{N}{2}$-point DCT-II matrix, expressed:

$$C_1 = k_m \begin{bmatrix} 1 & 1 & 1 & \cdots \\ 1 & cos(\frac{\pi}{\frac{N}{2}}) & cos(\frac{2\pi}{\frac{N}{2}}) & \cdots \\ 1 & cos(\frac{2\pi}{\frac{N}{2}}) & cos(\frac{4\pi}{N}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{5.28}$$

and

$$C_2 = k_m \begin{bmatrix} 1 & cos(\frac{1}{2}\frac{\pi}{N}) & cos(1\frac{\pi}{N}) & \cdots \\ 1 & cos(\frac{3}{2}\frac{\pi}{N}) & cos(3\frac{\pi}{N}) & \cdots \\ 1 & cos(\frac{5}{2}\frac{\pi}{N}) & cos(5\frac{\pi}{N}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{5.29}$$

From (5.25), it is necessary to take the DCT-I from $n = 0$ to $\frac{N}{2}$, yielding $\left(\frac{N}{2}+1\right)$ evaluation points rather than $\left(\frac{N}{2}\right)$ evaluation points. Hence, C_1 is has an extra row over C_2 and any input matrix must be padded with an extra column of zeros to accommodate for the matrix size.

Define a labeling of the spatial signal in (5.30), where an image signal X is divided into four sub-signals (its 2-D polyphase components): X_1, X_2, X_3, and X_4.

$$X = \begin{array}{|cc|cc|c|} \hline X_1(0,0) & X_2(0,0) & X_1(1,0) & X_2(1,0) & \\ X_4(0,0) & X_3(0,0) & X_4(1,0) & X_3(1,0) & \cdots \\ \hline X_1(0,1) & X_2(0,1) & X_1(1,1) & X_2(1,1) & \\ X_4(0,1) & X_3(0,1) & X_4(1,1) & X_3(1,1) & \cdots \\ \hline X_1(0,2) & X_2(0,2) & X_1(1,2) & X_2(1,2) & \\ X_4(0,2) & X_3(0,2) & X_4(1,2) & X_3(1,2) & \cdots \\ \hline & \vdots & & \vdots & \ddots \\ \hline \end{array} \tag{5.30}$$

From (5.30), we can determine four terms in (5.31).

$$\begin{aligned}
T_1 &= C_1^T \begin{pmatrix} X_1 & \mathbf{0} \\ \mathbf{0}^T & \end{pmatrix} C_1 + C_1^T \begin{pmatrix} X_1 \\ \mathbf{0}^T \end{pmatrix} C_2 + \\
&\qquad C_2^T \begin{pmatrix} X_1 & \mathbf{0} \end{pmatrix} C_1 + C_2^T X_1 C_2 \\
T_2 &= C_1^T \begin{pmatrix} \mathbf{0} & X_2 \\ & \mathbf{0}^T \end{pmatrix} C_1 + C_1^T \begin{pmatrix} X_2 \\ \mathbf{0}^T \end{pmatrix} C_2 + \\
&\qquad C_2^T \begin{pmatrix} \mathbf{0} & X_2 \end{pmatrix} C_1 + C_2^T X_2 C_2 \\
T_3 &= C_1^T \begin{pmatrix} & \mathbf{0}^T \\ \mathbf{0} & \\ & X_3 \end{pmatrix} C_1 + C_1^T \begin{pmatrix} \mathbf{0}^T \\ X_3 \end{pmatrix} C_2 + \\
&\qquad C_2^T \begin{pmatrix} \mathbf{0} & X_3 \end{pmatrix} C_1 + C_2^T X_1 C_2 \\
T_4 &= C_1^T \begin{pmatrix} \mathbf{0}^T & \\ & \mathbf{0} \\ X_4 & \end{pmatrix} C_1 + C_1^T \begin{pmatrix} \mathbf{0}^T \\ X_4 \end{pmatrix} C_2 + \\
&\qquad C_2^T \begin{pmatrix} X_4 & \mathbf{0} \end{pmatrix} C_1 + C_2^T X_4 C_2
\end{aligned} \tag{5.31}$$

Applying the matrices C_1 and C_2 once gives a 1-D DCT-I and DCT-II, respectively, where, for example $X_i C_2$ is the 1-D $\frac{N}{2}$ DCT-II of X_i. Applying them twice to X_i, e.g. $C_2^T X_i C_2$, gives the 2-D DCT-I and DCT-II of X_i. From [85], a combination of the four terms in (5.31) determines an $(N \times N)$ DCT in terms of $(\frac{N}{2} \times \frac{N}{2})$ DCTs.

$$X_{N\times N}^{C2}(l,m) = \hat{C}_2^T X \hat{C}_2 = k_l^T \cdot k_m \cdot \sum_{i=1}^{4} T_i \qquad \text{for } l,m = 1,2,\ldots \frac{N}{2} \tag{5.32}$$

In downsampling in each direction by 2, with reference to the image matrix, we will only retain one of the four X_i signals. For any X_i being the remaining matrix after decimation, the terms in the decimation in space equation as given in (5.32) can only be exactly determined for a single i, and the sum of $\{T_j : j \neq i\}$ determine the unknown information.

Given that our task is to determine high-resolution DCT coefficients from low-resolution DCT coefficients, our problem is exactly the reverse decimation problem, where $C_2^T X_i C_2$ is known for a single i and the remaining fifteen terms in (5.31) compose the recovery problem of (5.32). For simplicity, let our signal be known for $i = 1$ and let $\chi = C_2^T X_1 C_2$ denote our input.

Now, we can combine our algorithm with SVR to achieve the following relationship.

$$X_N^{C2}(l, m) = k_l^T k_m \{\chi + g(\chi) + SVR(\chi)\} \tag{5.33}$$

where χ and $g(\chi)$ are known or can be exactly determined, and $SVR(\chi)$ are the terms to be predicted. That is to say,

$$\begin{aligned} \chi &= C_2^T X_1 C_2 \\ g(\chi) &= C_2^T X_1 C_1 + C_1^T X_1 C_2 + C_1^T X_1 C_1 \\ SVR(\chi) &= \sum_i (\alpha_i^+ - \alpha_i^-) K(\chi, \chi_i) + b \end{aligned} \tag{5.34}$$

The $SVR(\chi)$ term in (5.34) predicts the label as defined by $\left(\sum_{j=2}^{4} T_j\right)$ from the feature χ, and the kernel matrix $K(\chi_i, \chi_j)$ represents a high-dimensional dot-product space and is optimized in the following section. The final solution is shown in Fig. 5.2.

5.5 Results and Analysis

We verify the SDP ($\mu_i \geq 0$) and QCQP problems in Sec. 5.2 by using an initial estimate for a single coefficient in five-fold cross-validation on 50 data points, achieving an optimal $C = 550$. The quantitative difference between the two programming problems is negligible (on the order of 10^{-8}), meaning that they are solving the same problem. Due to the small data set size, the cvx [54] Matlab toolbox result is shown in Fig. 5.3. Fig. 5.3, where $MSE = 3.6722 \pm 23.14$, depicts two examples of five-fold, regression runs on DC Coeffients are shown in Fig. 5.3 using 50 training points and 10 testing points.

Table 5.1 depicts the effect of learning a kernel matrix. With CVX [54], the values were obtained using five-fold cross-validation on fifty points and tested on

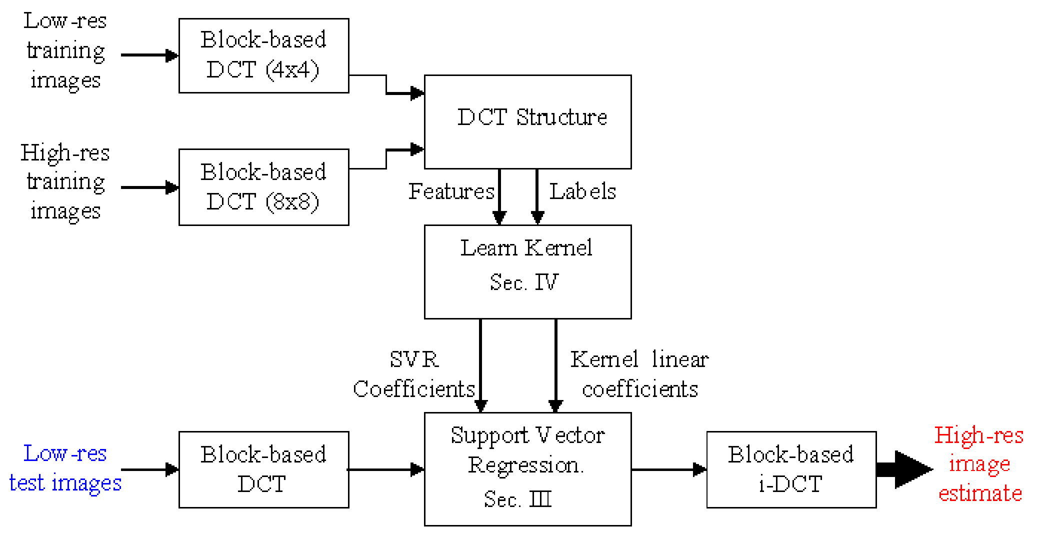


Figure 5.2: SVR Superresolution with DCT Structure

ten points in several domains as labeled in the data series column. We gave the option of three kernels and made use of the polynomial kernel $k_1(\mathbf{x}_1, \mathbf{x}_2) = (1 + \mathbf{x}_1^T \mathbf{x}_2)^d$ for K_1, a Gaussian kernel $k_2(\mathbf{x}_1, \mathbf{x}_2) = exp\left\{\left(-0.5(\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2)\right)/\sigma\right\}$ for K_2, and a linear kernel $k_3(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$ for K_3. "ZZ(i)" denotes the target series of the DCT output in zigzag scanned order, and "HR pix(i, j)" denotes the target series of the reconstructed high-resolution pixel values. The notation μ_i^+ means that we are dealing with the QCQP problem in which $\boldsymbol{\mu} \succeq 0$, and likewise, $\sum_i \mu_i K_i$ deals with the SDP problem. We benchmark using the optimally cross-validated Gaussian kernel matrix, which the fourth comparison labeled "Opt. C/V Gauss". The comparison mark uses the mean squared error (MSE) value, which is the sum of all the squared errors.

In the observations in Table 5.1, the cross-validation parameters suggest very wide Gaussian kernels (σ large) and high C values especially in the AC coefficients in the DCT domain, probably to accommodate for the variety of coefficients and pixel values. Another conclusion drawn from the table is that MSE values of DCT coefficients are inherently more difficult to predict than pixel values in the spatial domain. This fact can be explained by the ability, or tendency depending

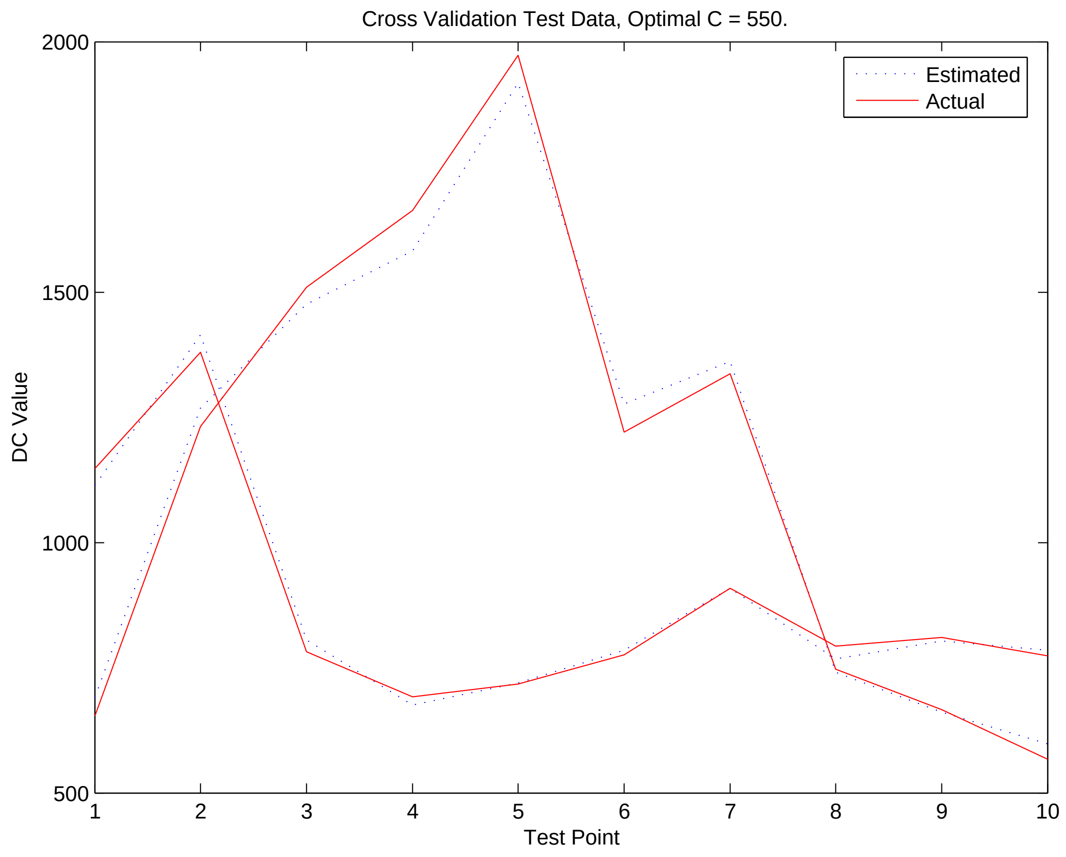


Figure 5.3: Regression Kernel Learning, Five Fold Cross-Validation

on its usage, of the DCT domain to collaterally de-correlate signals. In the end, if the metric is MSE, the solution from multiple kernels generally yields favorable results over optimally cross-validated Gaussian RBFs.

Experiments for actual image reconstruction in the superresolution problem were implemented with the MOSEK toolbox [1], which computed the QCQP problem in Sec. 5.2, alleviating the problem of the high complexity inherent in the SDP problem. In multi-dimensional input spaces involved, the degrees of freedom is significantly augmented by the choice of features for kernel selection. For example in the pixel domain where $D = 5$, there were $2^{(D \times D)}$ possible combinations to choose from for feature selection. We chose the most relevant combinations (i.e. the features surrounding a center pixel in a certain orientation.) For purposes of

Table 5.1: Training and Testing of Optimal Kernels

Data Series	K_1	K_2	K_3	Opt. C/V Gauss	$\sum_i \mu_i K_i$	$\sum_i \mu_i^+ K_i$
HR Pix (1,1)						
MSE	3,263.6	734.7	3,773.97	395.98	194.71	103.58
C/V Params	$d = 2$ $C = 700$	$\sigma = 100$ $C = 700$	C = 700	$\sigma = 400$ $C = 2,000$	C = 700	C = 700
$\mu_1/\mu_2/\mu_3$,	3/0/0,	0/3/0,	0/0/3,	-,	0.566/2.434/0,	0.6751/1.0505/1.2744
HR Pix (2,2)						
MSE	3,631.1	523.72	3,434.83	359.31	188.32	109.5
C/V Params	$d = 2$ $C = 700$	$\sigma = 100$ $C = 700$	C = 700	$\sigma = 400$ $C = 2,000$	$C = 700$	$C = 700$
$\mu_1/\mu_2/\mu_3$,	3/0/0,	0/3/0,	0/0/3,	-,	0.1441/2.8448/0.0111,	0.0149/2.9844/0.0007
DCT ZZ(1)						
MSE	16,193.1	6,039.2	11,647.1	4,545.3	2,043.2	2,375.7
C/V Params	$d = 2$ $C = 2000$	$\sigma = 300$ $C = 2000$	$C = 2000$	$\sigma = 500$ $C = 6000$	$C = 2000$	$C = 2000$
$\mu_1/\mu_2/\mu_3$,	3/0/0,	0/3/0,	0/0/3,	-,	4.24/2.38/-3.62,	1.8/1.2/0.0
DCT ZZ(2)						
MSE	3,810.6	35,189.6	22,686.3	12,901.3	1,519.5	1,383.7
C/V Params	$d = 2$ $C = 10,000$	$\sigma = 5,000$ $C = 10,000$	$C = 10,000$	$\sigma = 10^4$ $C = 10^6$	$C = 10,000$	$C = 10,000$
$\mu_1/\mu_2/\mu_3$,	3/0/0,	0/3/0,	0/0/3,	-,	2.03/1.05/-0.08	0.532/0.000/2.468

simplicity, the experiments set up for superresolving entire images used Gaussian RBF kernels exclusively with varying σ values, though it may better suit further research in the area to use other types of kernels including polynomial, sigmoid, linear, Laplacian, etc.

Fig. 5.5 and Fig. 5.4 are the result of superresolving 4×4 to 8×8 blocks in the block-based DCT algorithm as described in Sec. 5.4 The training set consists of only about a few thousand relevant input vectors from the same video sequence because memory for (5.22) increases quadratically in terms of quantity of data points, limiting our implementation. DCT SVR works very well when the testing data is similar to the training set, with PSNR values as high as 33.83 dB. Both methods are very clear, but looking at the leaves on the bushes, the sign, and the horse in front of the bus, it's evident that the structural improvements in the DCT domain effect a very crisp result.

The drawback of DCT SVR (and not the case in the next chapter) is shown

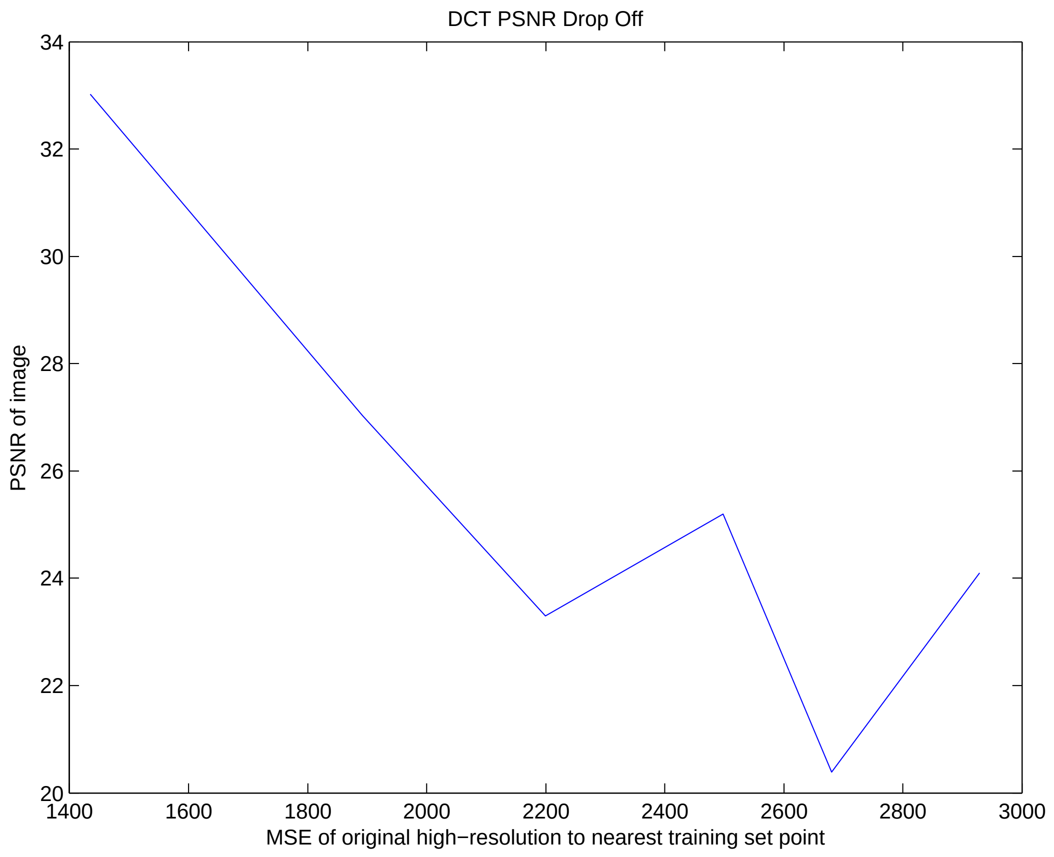
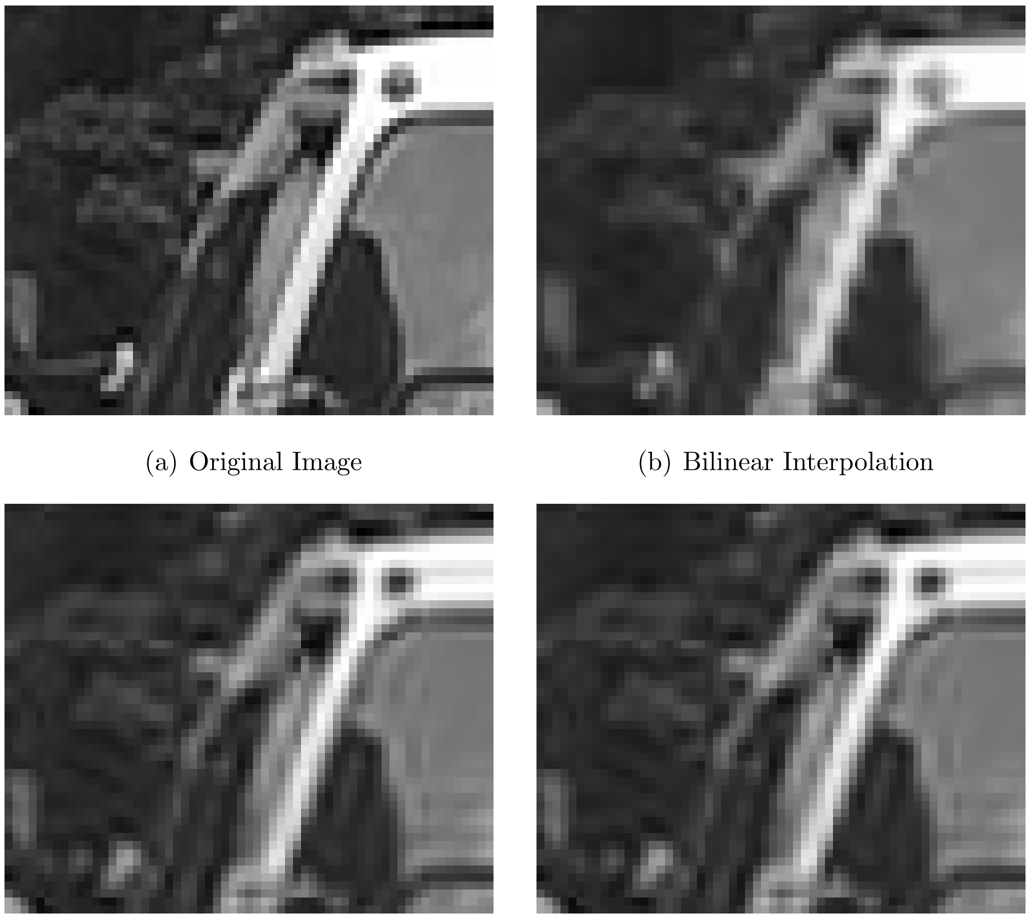


Figure 5.4: DCT Loss of Generalization

in Fig. 5.4, where it becomes clear that the training set needs to be larger in order to generalize well. This conclusion has been made by observing the sharp drop off in PSNR where comparison of like-images from test to training set show that the "further" away the test image is from the training set, the lower the PSNR value. Therefore, although the image appears very clear in Fig. 5.5, the idea could stand to improve. As will be seen in the next chapter, a mixture of experts framework provides the necessary improvements to perform better than the DCT domain-based interpolation.

Additional results of DCT domain-based SVR interpolation are reserved for the next chapter, where comparisons between techniques discussed in this chapter are compared with a SVR mixture framework.



(a) Original Image (b) Bilinear Interpolation

(c) Spatial Domain SVR (d) Frequency Domain SVR

Figure 5.5: Zoomed Reconstruction of 10th CIF "Bus" Sequence Frame

5.6 Summary

Given a training set, we solve the SVR problem for image interpolation while simultaneously optimizing the kernel function. Kernel optimization is achieved by minimizing with respect to a sum of smaller kernel matrices and reformulating the SVR QP as a SDP and QCQP forms. After programming problem formulations have been derived, it is possible to introduce structure into the domain to aid the learning process. For better regression, the DCT domain adds structure to the problem, although its ability to generalize is severely limited. Poor performance stems from the fact that small perturbations in AC coefficients result in large disturbances in the HVS. Such issues and several new ones shall be addressed and resolved in the next chapter where we introduce a new framework.

5.7 Acknowledgements

The text of Chapter 5 is adapted from *Support Vector Regression based Superresolution*, Karl Ni and Truong Nguyen, in the June 2007 issue of the *IEEE Transactions on Image Processing.* Portions of the chapter also appear in "Single Image Superresolution Based on Support Vector Regression", Karl Ni, S. Kumar, N. Vasconcelos, and Truong Nguyen, in Proceedings of the *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2006; and also in "Learning the Kernel Matrix for Superresolution", Karl Ni and Truong Nguyen, in Proceedings of the *IEEE* 8^{th} *International Workshop on Multimedia Signal Processing*, August 2006. The dissertation author was the primary author of these publications, and the listed co-author directed and supervised the research that forms the basis for this chapter.

6 Mixture of Experts for Image Interpolation

As Machiavelle taught them, divide and ye govern.

—1732 Swift Peoms III. 805

Simplifying regression structure by using structural properties of the DCT domain performs especially well when the training set is extremely large or is very similar to the testing image. This is quite often not the case as can be observed in Fig. 5.4 in Sec. 6.3 of the previous chapter, where PSNR values for video frames dissimilar to the training set drop off sharply as video frame temporal distance increases. As explained in the previous chapter, the HVS will detect noticeable differences should perturbations and slight errors in the energy of AC coefficients occur. While the effect can be mitigated to some extent by considering other domains (e.g. the wavelet domain [51]), the cost for currently standardized discrete-time domain changes[1] is poor generalization for gains in structural improvements of input domain selection. Moreover, with some thought, we may even accommodate for the advantages in domain structure with the inclusion of specialized kernels in the kernel learning process in Chapter 5.

Alternatively, classification-based algorithms [24, 9] have been shown to provide good generalization. One particularly successful classification-based framework is called a *mixture of experts* [61, 75], in which several function estimators, or

[1]The domain changes referred to here consist *only* of the DFT, DCT, and DWT. We have tested using only the Gaussian kernel. The final chapter discusses possible directions in research with respect to types of kernels.

"experts", pool their decisions together to vote on the most likely outcome. The final solution is a weighted combination, or "mixture", of each "expert's" belief. When the "experts" are defined as SVR estimators, then we have a *support vector mixture* [69, 43, 93].

The mixture of experts framework for image interpolation has been initially studied in [24], although the formulations of the "experts" as *nonlinear* regression are only superficially covered with the mere brief mention of the potential of such algorithms. This chapter is devoted to the development of a support vector mixture framework using SVR to provide image interpolation through a sliding-window approach. We further extend the application to colored images using novel techniques and properties of the HVS.

The following section, Sec. 6.1, highlights the proposed algorithm with more conventional definitions, and Sec 6.2 investigates the adaptation of the algorithm to the multiple color schemes in video transmissions, identifying which features to use for the least amount of estimation errors. Finally, Sec. 6.3 presents several simulation results.

6.1 Algorithmic Description

This section describes the hierarchical support vector mixture in detail. Sec. 6.1.1 relates the overall framework to conventional terminology. As an integral part of the stochastic framework, unsupervised classification, or clustering, is discussed in Sec. 6.1.2. Given probability metrics from Sec. 6.1.2, subsequent summarizations of convex optimization problems from Chapter 5 are tailored to each individual class in Sec. 6.1.3. Using a full mixture of SVR's, Sec. 6.1.3 fits discussed concepts into an overall framework to effect good image interpolation results.

6.1.1 Hierarchical Mixture of Experts

Diversity and complicated structures within images require much more than a single regression for all possible image content, even with scalable estimation techniques such as SVR. While SVR is ideally able to fit an arbitrarily complex system by properly cross-validating, without intimate knowledge of all the high-dimensional manifolds and subspaces of images (which involves a complex mixture of edges, gradients, texture, etc.), a perfect reproducing kernel Hilbert space (RKHS) using known kernel functions for a single SVR does not make sense. Consequently, the sheer number of parameters for an SVR with an imperfect kernel space renders the optimization problem unmanageable. Thus, instead of a single SVR, the central idea of the proposed algorithm is to use *multiple* SVR for generalization purposes.

While not explicitly stated in [88], the framework of the proposed algorithm is most adequately described as a mixture of experts [61] for regression, where given the same input vector, "experts" pool their decisions to vote on whose outcome is most likely correct. Systems of mixtures of experts typically use a feedforward network where all experts receive the same input, perform their processing, and the output is a weighted sum of the results, seen in (6.1).

$$f(\mathbf{x}) = h\left(\sum_{j=1}^{C} w_j(\mathbf{x}) s_j(\mathbf{x})\right) \tag{6.1}$$

In (6.1), $w_j(\mathbf{x})$ is a function of the input weighting the j^{th} regression expert, $s_j(\mathbf{x})$, and there are a total of C different experts.

SVR's replace $s_j(\mathbf{x})$ in Mixture of Support Vector Machines. Prior to SVR regression in $s_j(\mathbf{x})$, unsupervised clustering [48], as opposed to discriminant analysis, is denied access to any labels. Applying labels would require an inordinate amount of time because humans must assign classes to every training point. Four of the most commonly used clustering techniques [55] are exclusive, overlapping, hierarchical, and probabilistic clustering. Of the four types, only probabilistic clustering offers uncertainty measures of random patterns belonging to certain clusters. The attribute fits especially well because we will eventually require the use of posterior probabilities for both the weights $w_j(\mathbf{x})$ and regression $s_j(\mathbf{x})$ in (6.1). The uncertainty measures for a class, i.e. posterior probabilities $P(J = j|\mathbf{x})$, provide exactly that information, and the accuracy of the posteriors depends on the model used for classification and clustering.

When the training set Ω consists of N training pairs $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i$ is the i^{th} low-resolution training vector and $\mathbf{y}_i$ is the i^{th} high-resolution training vector, the estimation of function f in $\mathbf{y} = f(\mathbf{x})$ with $g(\mathbf{x})$ is given as the expected value, $g(\mathbf{x}) = E[\mathbf{y}|\mathbf{x}, \Omega]$. To integrate classification, the conditional expectation is found by $E[\mathbf{y}|\mathbf{x}] = \sum_j E[\mathbf{y}|\mathbf{x}, J = j]P(J = j|\mathbf{x})$ and thus,

$$g(\mathbf{x}) = \sum_{j=1}^{C} g_j(\mathbf{x})P(J = j|\mathbf{x}) \tag{6.2}$$

Here, the random variable J denotes a potential class of input vector $\mathbf{x}$.

Therefore, (6.2) equates an all-encompassing function $g(\mathbf{x})$ to averaging the result of several smaller functions $g_j(\mathbf{x})$ that sharpen, smooth, etc., depending on when it is necessary to do so. The posterior probability measures, $P(J = j|\mathbf{x})$, weight the regression results by how strongly the algorithm believes that a particular regression correctly represents the original function $f(\mathbf{x})$.

6.1.2 Unsupervised Classification

Clustering analysis is distinguished from discriminant analysis, the difference being that clustering is denied access to any labels, which means that it is categorized as an unsupervised learning technique [48]. There are different types of clustering where four of the most commonly used are [55]:

1. Exclusive Clustering
2. Overlapping Clustering
3. Hierarchical Clustering
4. Probabilistic Clustering

Of the four types, only probabilistic clustering offers uncertainty measures of random patterns belonging to certain clusters. Since we concentrate on hierarchical framework, probability measures are an attribute that fits especially well with (6.2) because 6.2 requires the use of posterior probabilities. Also, because the proposed algorithm eventually fits a nonlinear regression to training data points in a particular class, an important piece of knowledge would be which points to use for a single class regression and how likely it is that those particular points are in the class. The aforementioned uncertainty measures for a class provide exactly that information, the accuracy of which depends on the model.

The most common probabilistic clustering method uses GMM. As is the case in all model-based approaches, a mixture of parametric distributions (such as Gaussians in the case of GMM) models the entire training set. Each distribution represents a cluster, and the EM algorithm [39] is used to estimate the relevant parameters. As GMM's consist of the sum of several Gaussians, the estimated parameters for the j^{th} cluster include the prior probability of the cluster π_j, the mean $\boldsymbol{\mu}_j$, and the covariance matrix Σ_j.

By determining parameters for individual Gaussian curves of the GMM, EM can determine the likelihood of a point being in a class, $P(\mathbf{x}|J = j)$. The posterior probability in (6.2) can then be found with Bayes' Law:

$$\begin{aligned} P(J = j|\mathbf{x}) &= \frac{P(\mathbf{x}|J = j)P(J = j)}{P(\mathbf{x})} \\ &= \frac{P(\mathbf{x}|J = j)P(J = j)}{\sum\limits_j P(\mathbf{x}|J = j)P(j = j)} \end{aligned} \tag{6.3}$$

At runtime given an input vector $\mathbf{x}_{test}$, the task becomes finding the Gaussian in the GMM with the highest posterior probability at $\mathbf{x}_{test}$. Whichever Gaussian has the highest posterior is the class to which $\mathbf{x}_{test}$ most likely belongs.

Meanwhile, the complexity involved in training every point for M regressors, where M is the number of classes, is huge as interior point methods that eventually solve our regression problem in the next section scale according to $O(M^2N^{2.5})$. The complexity promotes M separate, smaller training sets $\Omega_j \subset \Omega$ for individual j^{th} regressors. Therefore, to obtain the subsets for simpler, more solvable problems, we take the points that are most relevant to the regression by thresholding the data for which the posterior probability is the highest. Therefore, for the j^{th} class, if a training point $\mathbf{x}_i$ meets a preset threshold η_j, then it is included in the training of the j^{th} regression.

Note that the simplification rejects considerable amounts of data per class, and it becomes extremely important not to overfit the data because test points have a good chance of not falling within a given class. Nevertheless, if SVR can indeed predict the relationship between low and high-resolution (as it has been shown to be capable of), then the regression may be sufficient for less relevant points in a given class. Furthermore, through experimentation, it turns out that only a few classes at any given test point are chosen and used for reconstruction most of the time. The implication is that, for the test point $\mathbf{x}_{test}$, by multiplying $P_{j|\mathbf{x}}(j|\mathbf{x}_{test})$ with $g_j(\mathbf{x}_{test})$ in (6.2), we would maintain good accuracy by zeroing out test data that is irrelevant for a particular class anyway, leaving reconstruction for those classes which can accurately do so.

6.1.3 Fitting the Framework

In each class in (6.2), some training points are more "important" than others for a given regression $s_j(\mathbf{x})$. The proposed algorithm must prioritize "important" points while marginally considering "unimportant" points.

Weighting training points is analogous to the effect of choosing C in the original primal problem, equation (6.4), on the solution hyperplane.

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i (\xi_i^+ + \xi_i^-)$$

subject to

$$\begin{aligned} y_i - (\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b) - \epsilon &\leq \xi_i^+ \\ (\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b) - y_i - \epsilon &\leq \xi_i^- \\ \xi_i^-, \xi_i^+ &\geq 0 \end{aligned} \tag{6.4}$$

The larger C is, the more penalty is incurred for non-flat regression solutions, in effect restricting the freedom to closely fit the training data in the constraints. The impact of C comes from its weighting of slack variables. Slack variables can be weighted differently for every point in the training set, and intuitively, for the j^{th} SVR, multiplying all ξ_i^- with the corresponding posterior probability of a particular class, $P(j = J|\mathbf{x}_i)$, would produce the desired effect.

Consequently, letting $\mathbf{P}_j$ be a vector of probabilities of all the points belonging to a class, the final equations can be rewritten from Chapter 5 and Sec. 6.1.2 by multiplying the posterior probabilities. Final equations are shown in (6.5) and (6.6).

Weighted SDP Problem for the c^{th} SVR

$$\begin{aligned}
\min_{\boldsymbol{\mu},t,\lambda,\mathbf{q}_u^+,\mathbf{q}_l^-,\mathbf{q}_u^-} \quad & t \\
\text{s.t.} \quad & \begin{pmatrix} 2\sum_i \mu_i K_i & \boldsymbol{\gamma} \\ \boldsymbol{\gamma}^T & t - 2C\mathbf{P}_c^T(\mathbf{q}_u^+ + \mathbf{q}_u^-) \end{pmatrix} \succeq 0 \\
& trace(K) = c \\
& \sum_i \mu_i K_i \succeq 0 \\
& \mathbf{q}_u^+, \mathbf{q}_u^-, \mathbf{q}_l^- \succeq 0 \\
& \epsilon e + \mathbf{q}_u^+ + \mathbf{q}_u^- - \mathbf{q}_l^- \succeq 0
\end{aligned} \tag{6.5}$$

Weighted QCQP Problem for the c^{th} SVR

$$\begin{aligned}
\max_{\boldsymbol{\beta}^+,\boldsymbol{\beta}^-,p} \quad & 2\mathbf{y}^T\boldsymbol{\beta}^- - 2\epsilon\mathbf{e}^T\boldsymbol{\beta}^+ - cp \\
\text{s.t.} \quad & p \geq \boldsymbol{\beta}^- K_i \boldsymbol{\beta}^- \\
& \mathbf{e}^T\boldsymbol{\beta}^- = 0 \\
& 0 \preceq \boldsymbol{\beta}^+ + \boldsymbol{\beta}^- \preceq 2\mathbf{P}_c C \\
& 0 \preceq \boldsymbol{\beta}^+ - \boldsymbol{\beta}^- \preceq 2\mathbf{P}_c C
\end{aligned} \tag{6.6}$$

6.2 Color Image Superresolution

This section details the problem of superresolving color, its place in the proposed framework algorithm, and related issues such as chroma subsampling. Sec. 6.2.1 describes the adaptation of the proposed algorithmic framework to general color superresolution. Sec. 6.2.2 explains how the different transmission formats of color affect the input features.

6.2.1 Integrating Color into the Stochastic Framework

Color images are usually partitioned into three components. The simplest prediction technique is independent component interpolation. There are obvious disadvantages to this method, most notably the disregard for inherent correlation between components. One readily available remedy for this issue and those similar to it would be to use values from all three components to produce an evenly proportioned feature representative of all three components. However, using 5×5 windows in 3 different spaces means 75 dimensions, and estimation errors overcome whatever is gained by the added information.

Therefore, we need to maintain balance by trading off small feature vector size for a decent amount of quality information. As it turns out, in terms of the human visual system (HVS), changes in color Cr and Cb components are less detectable, and perceptual changes in luminance seem more important. In fact, all MPEG compression use a 4:2:0 resolution format, where luminance pixels outnumber either chrominance component by a factor of 4. As a result, many techniques that use RGB interpolation (including [9]) weight the importance of each component by their average proportion of luminance. The proposed algorithm is more direct in its approach and clusters luminance components only, disregarding color altogether. The rationale behind this thinking is that for purposes of image content recognition, particular objects may be tinted differently when the underlying texture as well as the transition of colors within the patch remain the same.

While clustering luminance components in $\mathbf{x}$, color regressions use a separate input $\mathbf{z}$ from a window of surrounding color components (either Cb or Cr). It is here that SVR has distinct advantages over the linear filtering used by most interpolation algorithms. By filtering edges, halos or odd-colored auras often appear along texture transitions and borders. This is due in part to the averaging of pixel values, which smoothing linear filters have a tendency to do. Because there exists a multitude of shades of colors between any two chroma values, averaging often produces these unnatural and strange colors. The proposed algorithm avoids these undesirable byproducts that typically plague linear algorithms because the choice of training set often excludes these in-between values. The exclusion leaves out the odd-looking colors by effecting texture transitions that occur naturally in the image pairs of the training set.

6.2.2 Chroma-subsampling and Superresolution

There is an extensive quantity of standard transmission formats for video with color, each requiring a different implementation due to the disparity of the domain. The most commonly used scheme, 4:2:0, can be found in all versions of MPEG, JPEG, and various other standards, where there are 4 luminance components to single Cb and Cr components. MPEG-4 contains options for higher quality color, available in 4:4:4 schemes.

The consequences of the disproportionate sampling of color versus brightness mandate alternatives to the support of input vectors in Cb and Cr color components. In other words, how large the surrounding area of a color component is used for the regression. Too large of an area would cause estimation errors from excess dimensionality while simultaneously reducing generalization ability. Too small of a support area makes cross-validation difficult because insufficient discriminating information is present.

In the end, it makes sense for the Cb and Cr regression input vector support to cover the same area image-wise as the luminance regression rather than pixel-wise. That is to say, color components match the area of the luminance components by the subsampling factor. For example, in 4:2:0 schemes, a 5×5 luminance regression window would correspond to a 3×3 chrominance window where the surrounding chrominance window is half the extension of luminance pixels.

6.3 Results and Analysis

The algorithm clustered on 3×3 windows. Luminance and chrominance regression features come in 4:2:0 color schemes from windows of sizes 7×7 and 3×3, respectively. We ensure good generalization by randomly selecting a test image that differs from a training set of thirty diverse images from the Calphotos collection [23].

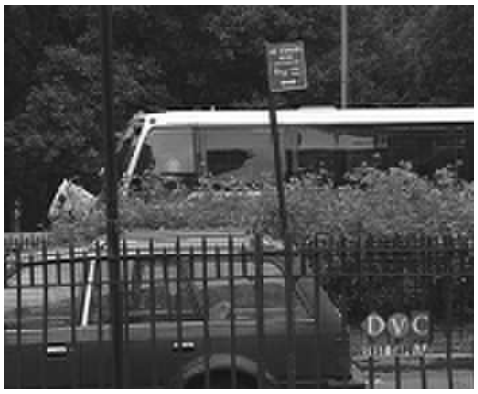
To generate the low-resolution images, high-resolution training images are filtered using an 11×11 anti-aliasing filter, specified by MATLAB's `imresize.m` function. Then, the image is downsized by $u = 2$ in each direction to obtain the low-resolution images. Every possible patch except for the edges (i.e. no zero-padding) are then taken from the low-resolution images, and the four high-resolution pixels corresponding to the center pixel of the low-resolution patch are used to complete the pair $(\mathbf{x}_i, \mathbf{y}_i) \in \Omega$.

Experiments for actual image reconstruction in the superresolution problem are implemented in the spatial domain with the MOSEK toolbox [1], which computes the QCQP problem in Sec. 6.1.3, alleviating the problem of the high complexity inherent in the SDP problem. In multi-dimensional input spaces involved, the degrees of freedom is significantly augmented by the choice of features for kernel selection. For example for 7×7 window sizes in regression, there are $2^{(7\times7)}$ possible combinations to choose from for feature selection. We chose the features surrounding a center pixel in various orientations: vertical, horizontal, center 3×3, etc.. For purposes of simplicity, the experiments set up for superresolving entire images used Gaussian RBF kernels exclusively with varying σ values, though it may better suit further research in the area to use other types of kernels including polynomial, sigmoid, linear, Laplacian, etc.

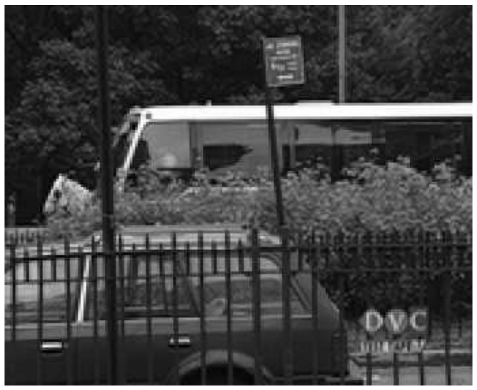
Comparisons superresolve by a factor of two. We compare to a few modern techniques, including a correlation-based approach [74], in which edges are sharpened or enhanced by preserving the low-resolution correlation matrix. Another edge-directed technique is subpixel edge localization [62], which also draws out edges by using geometrical relationships within a localized area of interest. Accompanying these comparisons are the linear counterpart to our approach, resolution synthesis [9], and the standard splines interpolation method, bicubic interpolation.

The results in Fig. 6.1 are based on a fairly small and slightly homogeneous training set. Using the simplified structures described in this chapter, we were able to increase our training set to 14 images in the CalPhotos image database from [23]. The images were chosen for content diversity, and so both quantity and quality of the training set are expanded.

(a) Original Image



(b) DCT Structured Reconstruction



(c) Mixture of Experts

Figure 6.1: Comparisons of DCT Domain to Mixture of Experts

In the proposed algorithm, the algorithm was set up with $D = 5$ and $U = 2$, meaning that I_{LR} was 5×5 and I_{HR} was 2×2, with clustering features of size 3×3 for 4 training images in the CalPhotos image database from [23]. For fair comparison, the same training set is used for any relevant learning algorithms involving a training set to which we compare the proposed algorithm. Note that by using an SVR-mixture and a considerably larger training set, the proposed algorithm has the ability to operate on a variety of different images, which is evident in the PSNR values for the bus video sequence in Fig. 6.5.

The techniques are compared quantitatively for frames in the bus sequence in Fig. 6.5 and qualitatively in Fig. 6.4 and Fig. 6.6. The proposed algorithm not only achieves more accuracy in PSNR than its linear counterpart in [9] and other referenced methods, visual comparisons offer better clarity in Fig. 6.4 and Fig. 6.6 as well. Because our algorithm performs image "estimation" as a purely mathematical optimization, high PSNR values are concentrated on first while allowing visual acuity to follow, whereas other algorithms in Fig. 6.5 have the objective to enhance visual acuity directly. In essence, numerically speaking, SVR obtain a more "correct" result.

Fig. 6.2 shows the difference between nonlinear SVR and linear filtering. Observing both images, due to presharpening, there appears to be extra noise in the linear filtering case, whereas the SVR shows a more crisp superresolution result. In the linear case, it is the presharpening filter that produces this effect, in essence, overcompensating with the inadequacies of linear filtering instead of producing exact results. This is also noticeable in Fig. 6.6, where the stripes continue through the building top, overcompensating when it is unnecessary to do so.

Comparisons to Li's edge directed interpolation [74], are shown in comparison to the proposed algorithm in Fig.6.3 on the Barbara image. There is an aliasing-like effect in the where there is close proximity of edges and gradients (the striped portions of the shirt and pants). This could be a by-product of a two pass system in which Li considers only 2×2 features at a time in isolated horizontal-vertical and diagonal passes. Therefore, drawing from 3×3 features at a time, the proposed algorithm asserts that joint consideration is advantageous, and indeed, the kernel chosen for the class is always one that includes the 3×3 features (though this is the case with most classes).

Comparisons to simpler interpolation techniques are also shown in Fig. 6.4. Of these include subpixel edge localization [62], and the result enhances edges quite well, perhaps even better than [74]. The side effect, though, is that the result looks slightly cartoonish. Fig. 6.4 also gives the inferior bicubic splines interpolation result. While these algorithms perform poorer than the proposed algorithm, it is worth it to note that they also have very little extra information to aid them, which is part of the reason why SVR reconstruction is superior in terms of quality.

Table 6.1: Miscellaneous PSNR comparisons

METHOD	**PSNR Values**		
	Jellybean	Chewbacca	Foreman,3
Bicubic	20.65 dB	17.11 dB	18.21 dB
NEDI [74]	19.33 dB	16.40 dB	17.90 dB
SEL [62]	12.18 dB	15.73 dB	18.34 dB
Linear RS [10]	19.69 dB	16.94 dB	17.81 dB
Proposed Algorithm	20.57 dB	18.24 dB	18.60 dB

The techniques are shown quantitatively in Table 6.1. The results show that the proposed algorithm is able to yield higher PSNR values than contemporary, competitive algorithms on average. One non-intuitive result in the quantitative comparisons are bicubic interpolation's relatively high PSNR values. This can be somewhat explained by MATLAB's half-band filter after upsampling, in which the inherent information never deviates too far from the actual values. Numerically speaking, averaging surrounding pixels creates a somewhat "safe" result, never deviating too far from the true pixel value, but also, never giving a crisp result. While numerically superior, in the end, the visual results in Fig. 6.7, Fig. 6.8, and Fig. 6.9 speak for themselves.

All qualitative comparisons are shown in Fig. 6.7. Already, some of the fur texture has been lost in all methods except for the proposed algorithm, due to the specificity of the functions $g_j(\mathbf{x})$. Even the linear classification method [9] in Fig 6.7(e) is inadequate in this respect, albeit the severity is to a much lesser degree. Further examining the difference between linear and nonlinear regression is Fig. 6.9. Observing both images, due to presharpening, there appears to be extra noise and pixelation in the linear filtering case, whereas SVR shows a more crisp superresolution result. This result can be attributed to the presharpening filters overcompensating for the inadequacies of linear filtering instead of producing exact results.

Additional visual comparisons to current interpolation techniques are shown Fig. 6.8. The errors in Li et. al [74] may be a by-product of a two pass system in which Li considers only 2×2 features at a time in isolated horizontal-vertical and diagonal passes. Therefore, drawing from 3×3 features at a time, kernel resolution synthesis asserts that joint consideration is advantageous, and indeed, the kernel chosen for the class is always one that includes the 3×3 features (though this is the case with most classes). [62] is a somewhat simpler technique, and the result enhances edges quite well, perhaps even better than [74]. The side effect, though, is that the result looks slightly cartoonish. Fig. 6.7 also gives the inferior bicubic splines interpolation result. While these algorithms perform poorer than the proposed algorithm, it is worth it to note that they also have very little extra information to aid them, which is part of the reason why SVR reconstruction is superior in terms of quality.

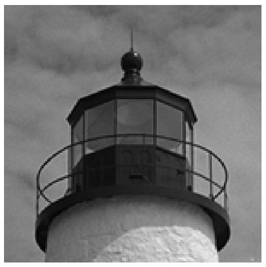
6.4 Summary

Through a classification-based, mixture of experts frameworks, problem complexity has been reduced and better generalization is offered. The effect of the improvements on the image interpolation problem has produced results better in both visual acuity and PSNR values. Regression in the color domain based on luminance classification works very well. Correctly tuned, the ill-effects of bleeding into surrounding edge areas will not occur in this scheme.

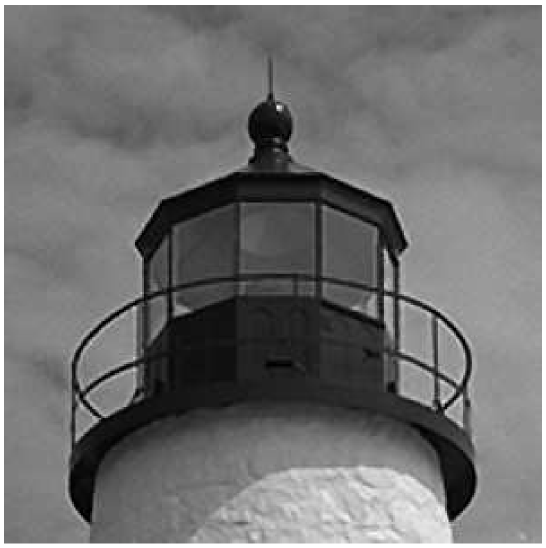
6.5 Acknowledgements

The text of Chapter 6 is adapted from *Mixture of Experts Framework for Superresolution*, Karl Ni and Truong Nguyen, January 2008, manuscript submitted to the *IEEE Transactions on Image Processing.* Portions of the chapter also appear in “Kernel Resolution Synthesis for Superresolution”, Karl Ni and Truong Nguyen, in Proceedings of *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 2007; “Complex Function Estimation using a Stochastic Classi-

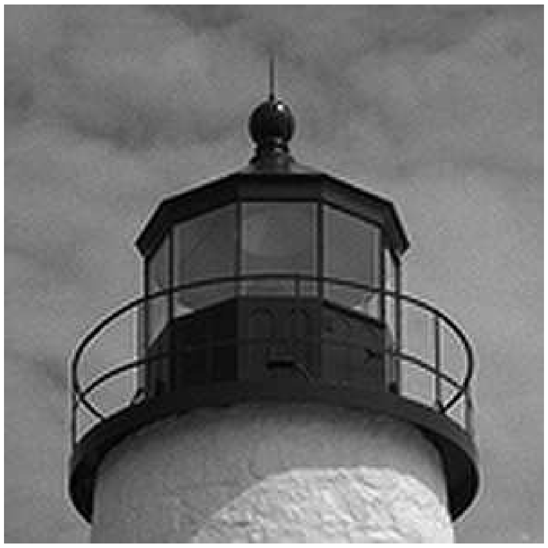
fication/Regression Framework: Specific Applications to Image Superresolution", in Proceedings of the *SPIE International Conference on Optics and Photonics*, August 2007; and also in "Color Image Superresolution Based on a Stochastic Combinational Classification-Regression Algorithm", in Proceedings of the *IEEE International Conference on Image Processing*, September 2007. The dissertation author was the primary author of this publication, and the co-author listed directed and supervised the research which forms the basis for this chapter.



(a) Original



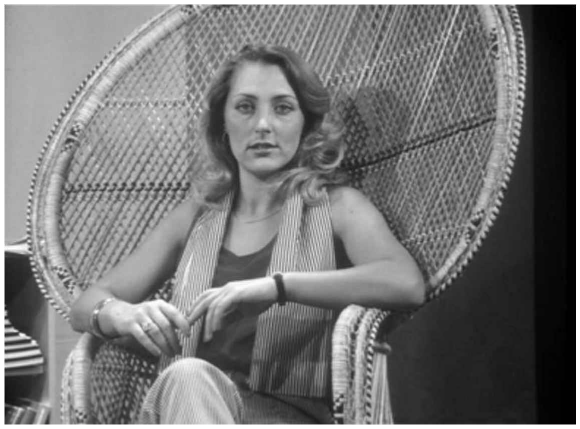
(b) Proposed Algorithm



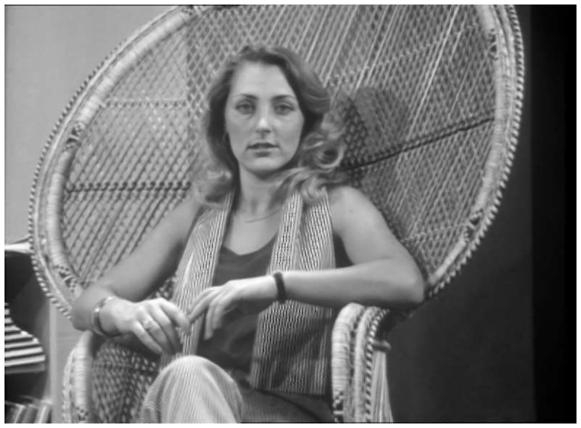
(c) Resolution Synthesis [9]

Figure 6.2: Proposed Algorithm versus Resolution Synthesis [9]

(a) Original Image

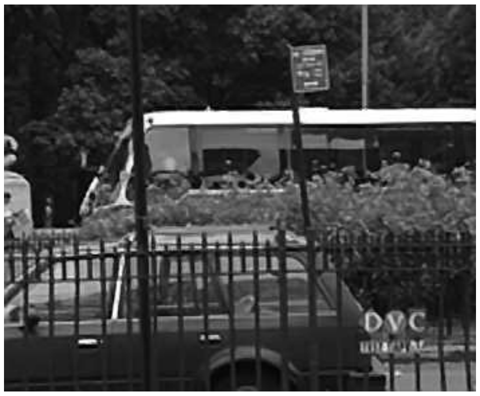


(b) Proposed Algorithm

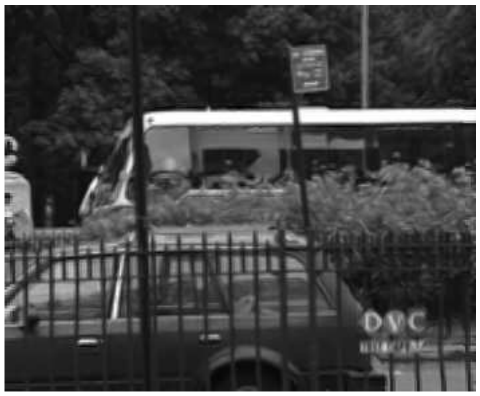


(c) NEDI [74]

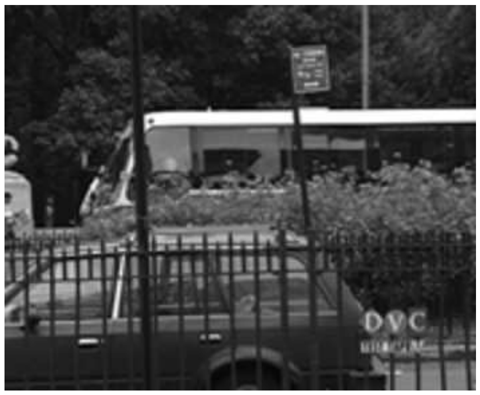
Figure 6.3: Proposed Algorithm versus Edge Directed Interpolation [74]



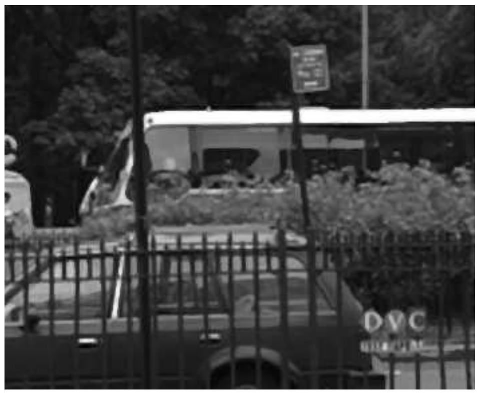
(a) Proposed Algorithm



(b) NEDI [74]



(c) Bicubic Interpolation



(d) Subpixel Edge Localization [62]

Figure 6.4: Proposed Algorithm vs Various State of the Art Algorithms

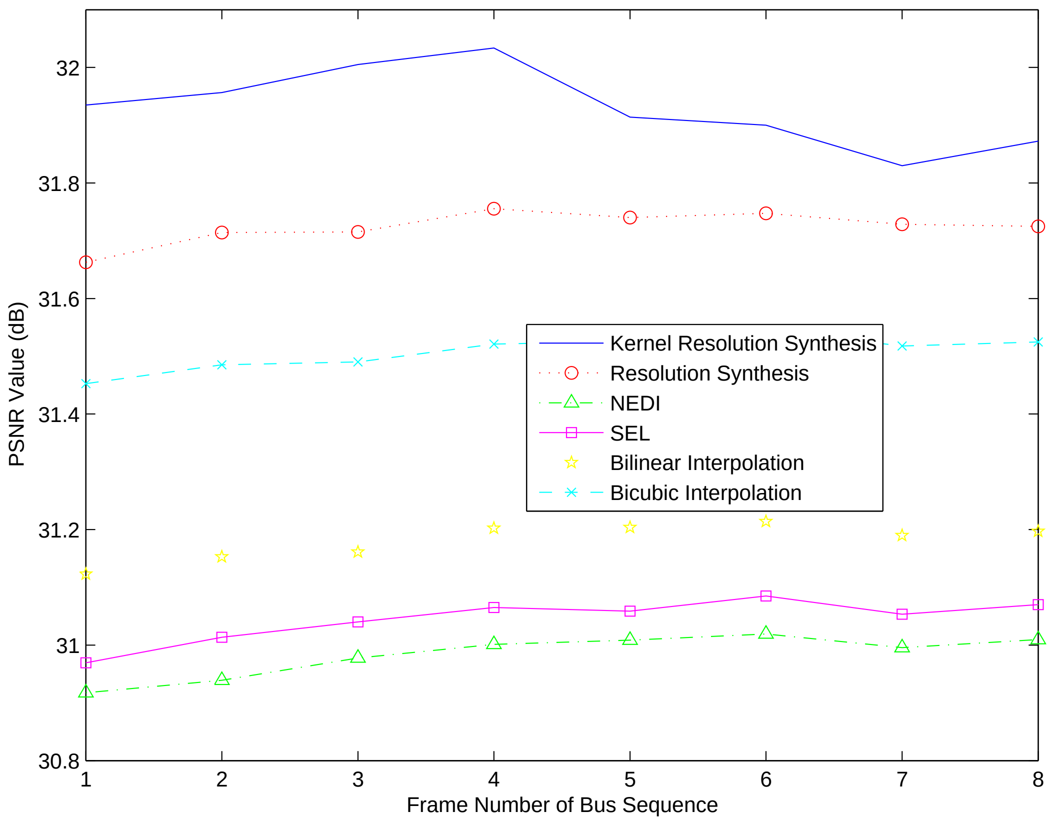
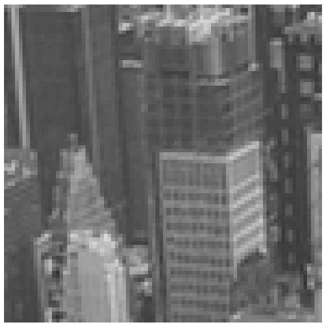
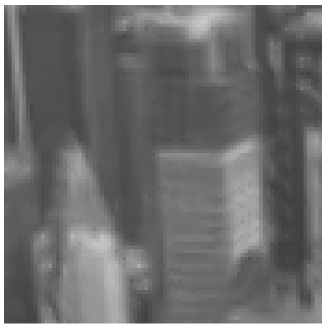
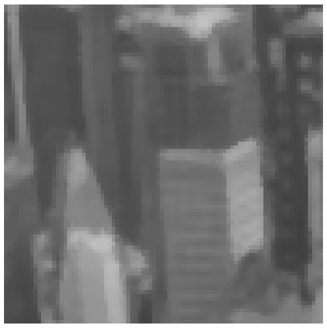
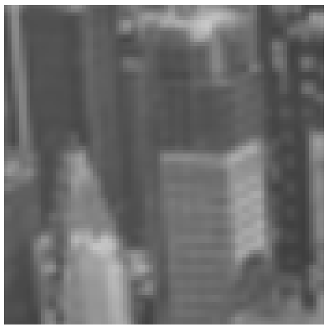


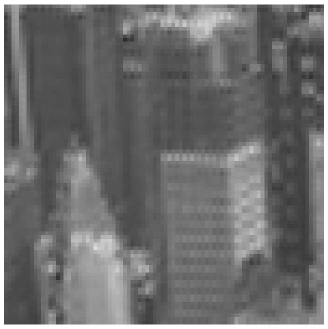
Figure 6.5: PSNR Results in 8 Frames of the Bus Sequence.


(a) Original Image


(b) NEDI [74]


(c) SEL


(d) Bicubic


(e) Resolution Synthesis

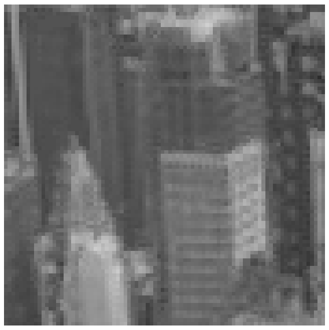
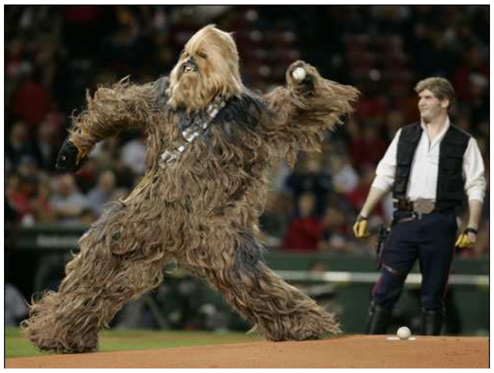
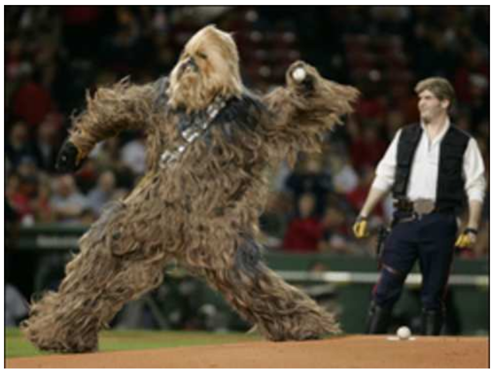

(f) Support Vector Mixture

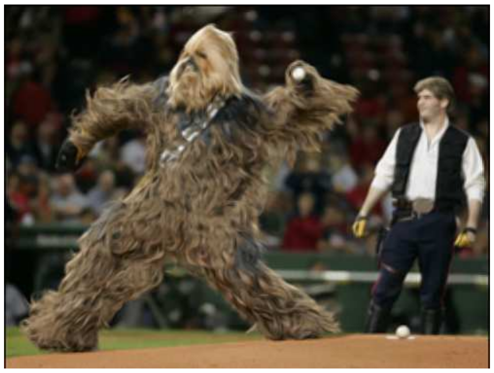
Figure 6.6: Comparisons to State of the Art, 6th Frame City Sequence



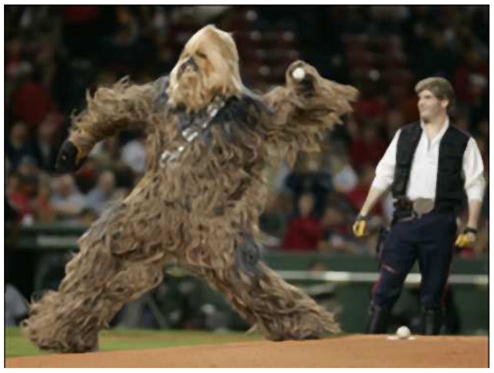
(a) Original



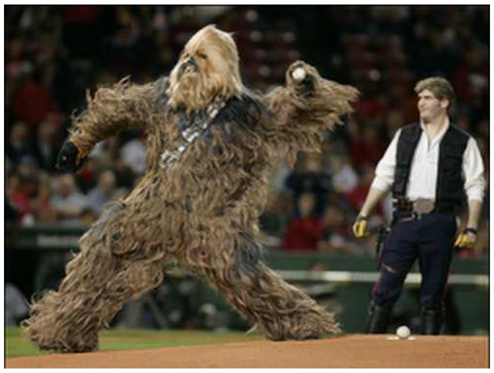
(b) Bicubic Interpolation



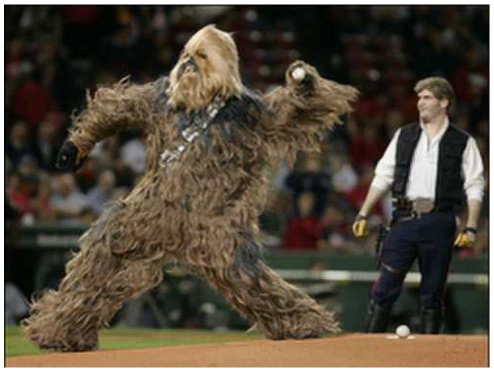
(c) Edge Directed Interpolation



(d) Subpixel Edge Localization

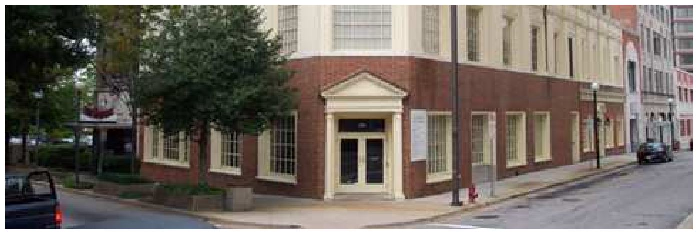


(e) Resolution Synthesis

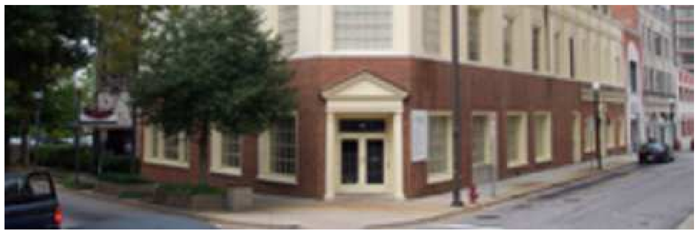


(f) Mixture of Experts

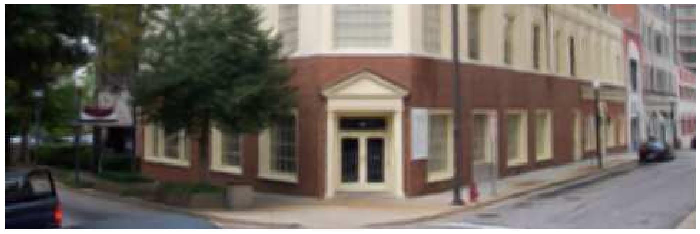
Figure 6.7: Comparisons to State of the Art



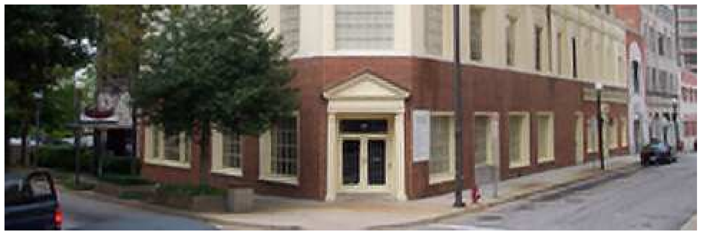
(a) Original



(b) Bicubic Interpolation

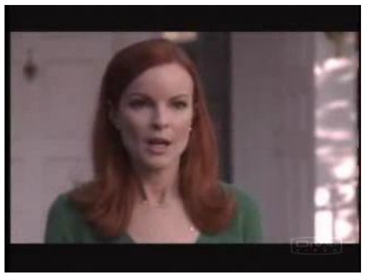


(c) Edge Directed Interpolation [74]

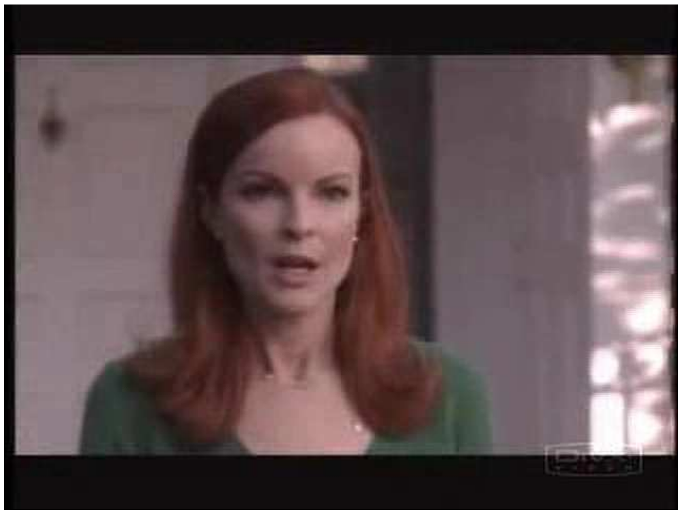


(d) Mixture of Experts

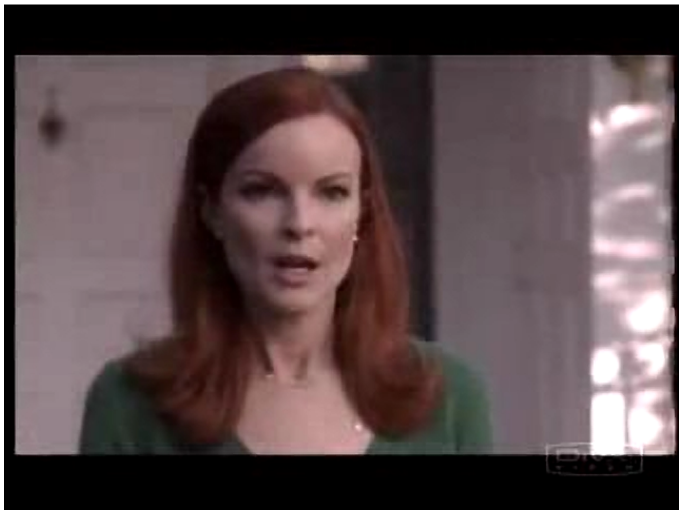
Figure 6.8: Comparisons to State of the Art Interpolation Techniques



(a) Original



(b) Resolution Synthesis



(c) Mixture of Experts

Figure 6.9: Mixture of Experts vs C.B. Atkins [9]

7 Polyphase Representation of Classification-Based Filtering for Image Interpolation

In order to change an attitude, then, it is presumably necessary to modify the information on which that attitude rests.

—Richard Petty and John Cacioppo

Results of image construction efforts in the previous chapters are impressive, but the machinery required for the nonlinear interpolation algorithms currently do not justify their use in commercial products. The best tradeoff between quality and computational complexity has been discussed in Chapter 2, and of the cited related works, we have found C. B. Atkins' Ph.D. thesis [9] to be the most viable, currently already implemented in newer HP printer models (See United States Patent 7149369, European Patent EP0874330). Because [9] (and other closely related algorithms [24, 56]) forgo frequency analysis, analytic filter design, or signal analysis, we discuss several areas of expansion while maintaining their empirical design aspect. It is our intention to explore [9] using such tools to reach more intuitive conclusions and propose some novel ideas.

On the opposite end of the spectrum, up until now, we have used pattern classification as an instrument in solving specific signal processing problems. The remainder of this thesis observes the converse: signal processing analysis of an interpolation method that has primarily been developed in the machine learning sense. Sec. 7.1 briefly reviews [9] and discusses the algorithm as it is currently interpreted. Using polyphase analysis, Sec. 7.2 interprets [9] as a weighted mixture of filters and expresses the approach in terms of filterbanks. Sec. 7.3 proposes the comparable performance of a single, zero-phase filter rather than a collection of filters to furthermore reduce computational complexity. Finally, we propose to generalize filter design to arbitrary scaling factors while simultaneously keeping complexity low in Sec. 7.4.

7.1 Review: C. B. Atkins' Ph. D. Dissertation

Let Ω equal $\{(\mathbf{x}_i, \mathbf{y}_i)\}$, a training set consisting of T pre-processed input/output pairs. Our terminology of an $\mathbf{x}$ and $\mathbf{y}$ that denote low and high-resolution patches, respectively, is opposite of most superresolution papers [96, 17], which study the reverse problem. Because filter coefficients are derived from the training pairs directly, the most noticeable of the few differences is the MMSE expression, which will become clearer towards the end of this section.

To interpolate an image by u in both horizontal and vertical directions, algorithms slide a $d \times d$ window across the low-resolution image. Processing a given $d \times d$ image patch requires vector representation $\mathbf{x} \in \mathbb{R}^{d^2 \times 1}$. The resolution enhancement at the center pixel of the $d \times d$ patch corresponds to $u \times u$ pixels in the high-resolution image. Therefore, $\mathbf{y}$ has dimensions $u^2 \times 1$.

The bulk of [9] is a collection of two journal papers, defining the proposed algorithms as "resolution synthesis" in [12] and "tree-based resolution synthesis" [11], the latter of which has evolved from the former. As the title of [9] suggests, classification is used as a mechanism for content-specific regression. That is, if j represents a particular class number, both "resolution synthesis" [12] and "tree-based resolution synthesis" [11] find the best j for a given $\mathbf{x}$ and then provide the regression seen in (7.1). The frameworks differ in that "tree-based resolution synthesis" relies on binary classification, using the first principle component and the mean to divide training points, while resolution synthesis assumes a GMM, using individual Gaussians of the mixture to represent a single class.

In both [12] and [11], the underlying equation from which parameters are to be estimated is given in the linear equation in (7.1).

$$\mathbf{y} = A^{(j)}\mathbf{x} + \beta^{(j)} \tag{7.1}$$

where $A^{(j)}$ is a matrix of size $d^2 \times u^2$, and $\beta^{(j)}$ is a vector of size $u^2 \times 1$ for the class j. [12] and [11] determine $A^{(j)}$ and $\beta^{(j)}$ differently for all j, although ultimately, both use some type of weighted least-squares-like approach.

The expression that [9] associates with the training set relating domain to range is $E[\mathbf{y}|\mathbf{x}]$. [11] directly approximates $E[\mathbf{y}|\mathbf{x}]$ using (7.1). Consequently, design of $A^{(j)}$ and $\beta^{(j)}$ is relatively simple, and filter coefficients can be found by the least squares solution in

$$A^{(j)} = R_{xy}R_{xx}^{-1} \tag{7.2}$$

where autocorrelation and cross-correlation matrices, R_{xx} and R_{xy}, are maximum likelihood estimates. Likewise, using the maximum likelihood estimate of the means of a class, $\boldsymbol{\mu}_x^{(j)}$ and $\boldsymbol{\mu}_y^{(j)}$, $\beta^{(j)}$ is a normalization term defined by (7.3).

$$\beta^{(j)} = \boldsymbol{\mu}_y^{(j)} - A^{(j)}\boldsymbol{\mu}_x^{(j)} \tag{7.3}$$

Alternatively, [12], like previous works in image interpolation [88, 86], adopts a mixture of experts framework for interpolation. Recall in Chapter 6, the mixture of experts is commonly expressed as $\sum_j g^{(j)}(\mathbf{x})P(J = j|\mathbf{x})$, J being a random variable denoting the true class of $\mathbf{x}$. Substituting (7.1) for $g^{(j)}(\mathbf{x})$, the mathematical equivalent for linear regression sets $g^{(j)}(\mathbf{x}) = A'^{(j)}\mathbf{x} + \beta'^{(j)}$, and (7.4) is a weighted sum where posterior probabilities $P(J = j|\mathbf{x})$ come out of hierarchical clustering via EM.

$$\begin{aligned} E[\mathbf{y}|\mathbf{x}] &= \sum_j E[\mathbf{y}|\mathbf{x}, J = j]P(J = j|\mathbf{x}) \\ &= \sum_j \left(A'^{(j)}\mathbf{x} + \beta'^{(j)}\right) P(J = j|\mathbf{x}) \end{aligned} \tag{7.4}$$

The algorithm closely resembles a non-orthogonal, interdependent filterbank where the end summation has non-unity weights, which are determined by local image content. [12] makes full use of the posterior probabilities, $P(J = j|\mathbf{x}_i)$ in the training set by writing them in matrix form, P. As shown in (7.5), $A'^{(j)}$ and $\beta'^{(j)}$ are slightly different than that of (7.2) and (7.3).

$$\begin{aligned} A'^{(j)} &= (R_{yx} \cdot P)\,(R_{xx} \cdot P)^{-1} \\ \beta'^{(j)} &= \boldsymbol{\mu}_y^{(j)} - A'^{(j)}\boldsymbol{\mu}_x^{(j)} \end{aligned} \tag{7.5}$$

[9] calls the process of creating $A^{(j)}$ and $A'^{(j)}$ filter design. The term is technically correct, but for conceptualization, requires some manipulation in the way data is organized. Typically, image filters are represented as two-dimensional fixed kernels. Vectorizing, while standard in the pattern recognition, removes a level of intuition with respect to image processing. Moreover, it is difficult to relate to any image properties and see anything more than a linear system of equations without rewriting (7.1). In later sections, we focus on two-dimensional intuition in hopes of better generalization in application and understanding.

7.2 Polyphase Representation of Classification-based Filtering

Polyphase decomposition of signals [121, 111], originating from Bellenger et. al. [15], is the representation of the signals as a regularly multiplexed, parallel set of subbands. Polyphase representation has been fundamental to multirate applications, derivations of sampling theorems, uniform DFT filterbanks, and of interest to us, decimation and interpolation filters. Interpolation using polyphase [46] was first used for interpolation and decimation in 1-dimensional signals in a seminal paper by Valenzuela and Constantinidas [33]. The scope is broadened in several Vaidyanathan works [121, 122]. It is the concepts and analysis in [121] and [111] that catalyzes our contribution in improving our understanding of (7.1). The following subsections review key concepts and express (7.1) in polyphase form.

7.2.1 Review: Polyphase Decomposition

Let **h** be the impulse response of a digital filter of length N and $h[n]$ be the n^{th} filter tap, where $n = 0, 1, \ldots, N-1$. The transfer function of $h[n]$, $H(z)$, can be written as

$$H(z) = \sum_{n=0}^{N-1} z^{-n} h[n]. \tag{7.6}$$

We can organize (7.6) into M terms[1], where

$$\begin{aligned} H(z) = \; & \{h[0] + z^{-M}h[M] + \ldots\} + \\ & z^{-1}\{h[1] + z^{-M+1}h[M+1] + \ldots\} + \\ & \cdots + z^{-M+1}\{h[M-1] + z^{-1}h[2m-1] + \ldots\}. \end{aligned} \tag{7.7}$$

The m^{th} polyphase component of $H(z)$, $E_m(z)$ is defined as

$$\begin{aligned} E_m(z) &= \{h[m] + z^{-1}h[M+m] + \ldots\} \\ &= \sum_{n=0}^{\lfloor N/M \rfloor} z^{-n} h[nM+m]. \end{aligned} \tag{7.8}$$

Substituting (7.8), we can express (7.6) as

$$\begin{aligned} H(z) &= E_0(z^M) + z^{-1}E_1(z^M) + \ldots z^{M-1}E_{M-1}(z^M) \\ &= \sum_{m=0}^{M} z^{-m} E_m(z^M), \end{aligned} \tag{7.9}$$

and $H(z)$ is given as a sum of its polyphase components.

$H(z)$ is *symmetric* if $h[n] = h[N-1-n]$ and *anti-symmetric* if $h[n] = -h[N-1-n]$. If $H(z)$ is either symmetric or anti-symmetric, then it has *linear phase*. Linear phase filters have constant group delay, which means that all frequencies have constant delay times and phase distortion is avoided. Because the defining characteristics of images can be found in the phase [53], attributes of linear phase filters including constant group delay are favorable. A subset of linear phase filters to be discussed in Sec. 7.3 include *zero-phase* filters in which signal phase is unaffected.

[1]When we wish to interpolate in one dimension, $M = u$.

7.2.2 Class-specific Polyphase Filters

Recall that $A^{(j)}$ is a matrix of size $d^2 \times u^2$. We observe that each element in $\mathbf{y}$ comes about through a single row multiplication of $A^{(j)}$ with $\mathbf{x}$. That is, if $A^{(j)}_m \in \mathbb{R}^{d^2 \times 1}$ is the m^{th} row of $A^{(j)}$ denoting the j^{th} filter, then the m^{th} element of $\mathbf{y}$ is given by $y_m = A^{(j)}_m \mathbf{x}$. Each row of $A^{(j)}$ can be thought of as an individual filter to produce a single output value. Let us define $E^{(j)}_m$ as the two-dimensional kernel representation of $A^{(j)}_m$ shown in Fig. 7.1 when $u = 2$. (Obtaining $E^{(j)}_m$ is simply the reverse of whatever operation was used to vectorize $\mathbf{x}$.) Then, we can place the kernel $E^{(j)}_m$ around the $d \times d$ window to produce the filtered result y_m.

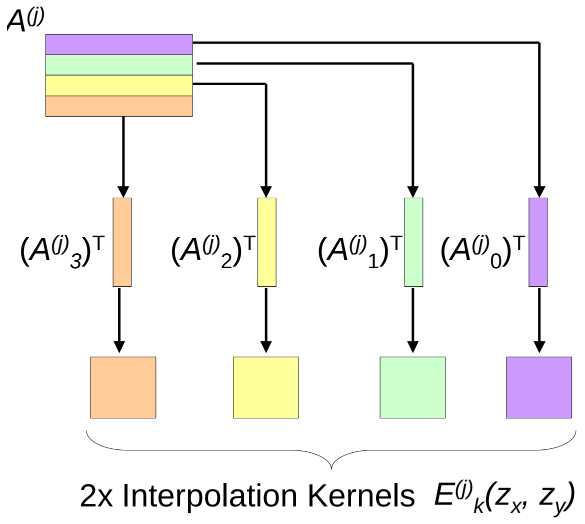


Figure 7.1: Two-dimensional Interpolation Kernels for Image Filtering

Our choice of nomenclature by using "E" in $E^{(j)}_m$ is not accidental; $E^{(j)}_m$ is the m^{th} polyphase component of a reconstruction filter. The high-resolution image is generated by placing y_m into the image after upsampling by u in horizontal and vertical directions. See Fig. 7.3(a) for details when $u = 2$.

We would like to express our filter bank collectively as a single filter. Thus, we require a simple application of the second of two *noble identities* [111], where

$$\textit{First Noble Identity}: \quad G(z)(\downarrow M) = (\downarrow M)G(z^M)$$

$$\textit{Second Noble Identity}: \quad (\uparrow M)G(z) = G(z^M)(\uparrow M) \tag{7.10}$$

The noble identities allow us to push the down/upsampling block through the filter and reverse the order of blocks, shown in Fig. 7.2.

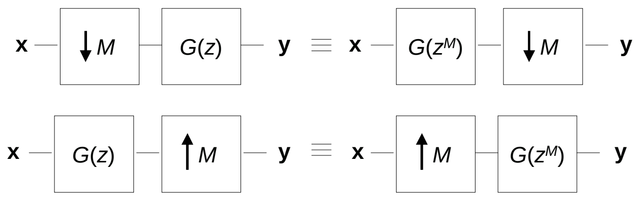
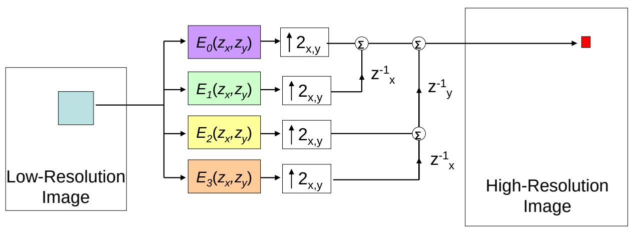
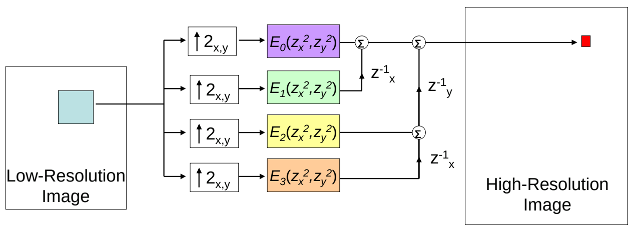


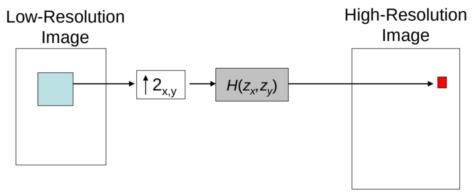
Figure 7.2: Noble Identities



(a) Polyphase Filters for Image Reconstruction.



(b) Use of Noble-Identities.



(c) Incorporation of all Polyphase Components into $H^{(j)}(z_x, z_y)$.

Figure 7.3: Interpolation Process for Class j.

With (7.10), we say that Fig. 7.3(a) and Fig. 7.3(b) are equivalent. We can then collapse Fig. 7.3(b) into a single filter by using the two-dimensional version of (7.9). The culmination is (7.11); an overall filter corresponding to the j^{th} class in an interpolating scheme where $u = 2$ is given by a sum of shifted polyphase components, which have been derived from $A^{(j)}$.

$$\begin{aligned} H^{(j)}(z_x, z_y) &= E_0(z_x^2, z_y^2) + z_x^{-1} E_1(z_x^2, z_y^2) + \\ &\qquad z_y^{-1} E_2(z_x^2, z_y^2) + z_x^{-1} z_y^{-1} E_3(z_x^2, z_y^2) \end{aligned} \tag{7.11}$$

Fig. 7.3(c) is the implementation of (7.11) and is equivalent, as well, to Fig. 7.3(a) and Fig. 7.3(b).

We have thus equated four independent filters to be the polyphase components of an overall filter, $H^{(j)}(z_x, z_y)$. In terms of [9], we can describe the entire interpolation procedure with the following steps:

1. Classify image block with the Frobenius Matrix Norm,
2. Upsample the two-dimensional signal by u,
3. Filter the upsampled signal with $H^{(j)}(z_x, z_y)$,
4. Determine the output signal by using the appropriate framework defined by either [11] or [12].

We have saved a few steps by avoiding vectorization, but more importantly, we can face the interpolation problem in two dimensions, which is how image problems are traditionally analyzed. Among the advantages stemming from two-dimensional intuition is 2-D Fourier-based analysis, which we utilize in subsequent sections.

7.3 Zero-phase Filter Design for a Single Interpolation Filter

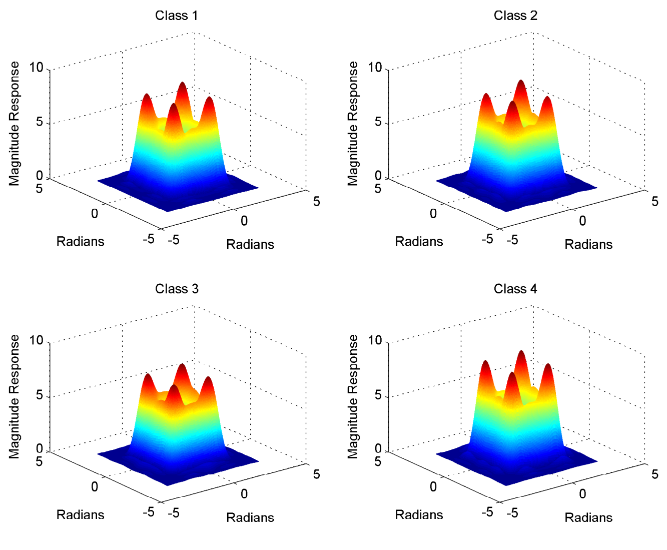
Despite the runtime improvement over nonlinear regression techniques in the previous chapters, the arguments against [9] do not preclude computational considerations. Classifying multi-dimensional vectors among a few hundred classes is a daunting task, and moreover, adopting any one framework to do so incurs non-trivial overhead. For example, with a mixture of experts [12], inputs are passed through a 100-tap filter for every class. Meanwhile, [11] iterates through a binary tree and needs $O(C \log C)$ evaluations plus comparisons, where C is the number of classes. Finally, vector quantization and nearest cluster evaluation [56] become especially expensive because they require C to be "large" as opposed to [9]. In the absence of CPU/GPU power or specialized hardware chips, the implementation may not support real-time video due to the toll that classification and the processing of multiple classes takes on the overall interpolative effort.

In the presence of complexity issues, a basic question to ask when evaluating class-specific filters is, how many classes should we use? Several papers with applications to biology have attempted to find a generic technique to determine the intrinsic structure of data [48, 18], specifically in the EM setting. Our experimentation on the matter[2] finds that the optimal number of classes varies for each test image and is surprisingly low, with PSNR relatively resilient to varying number of classes. This suggests some inherent similarities between individual class filters. In fact, using principles in Sec. 7.2 to create $H^{(j)}(z_x, z_y)$ for all $j = 1, 2, \dots, C$, we can observe some peculiar behavior in the Fourier domain that seems universal for all j and any C. In Fig. 7.4, we have zero-padded the impulse response, $h^{(j)}[n_x, n_y]$, out to 256^2 total samples (where before it was 10^2) to generate a smooth DFT curve[3] for $C = 4$ and $u = 2$. The trend in Fig. 7.4 is consistent for the different C with which we have experimented, where the differences between classes in the Fourier domain lie in the phase responses.

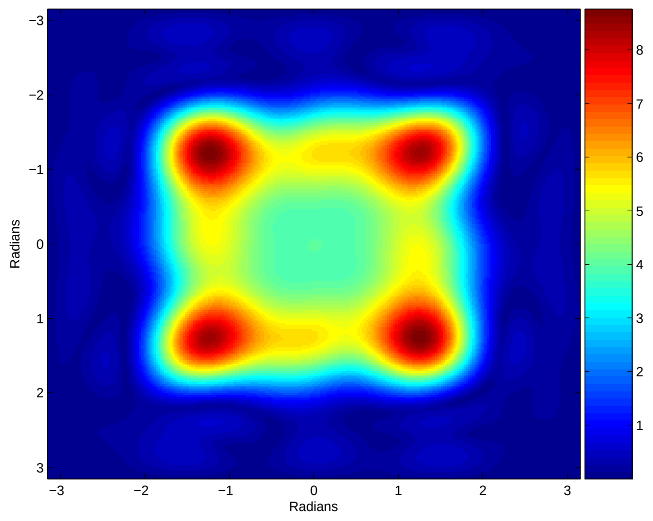
In lieu of recognizing patterns in the training set itself, we can try to make sense of the shape of the magnitude response. We describe the initial low-frequency sections from 0 to about $\frac{\pi}{3}$, or the center of Fig. 7.4(b), of any one dimensional slice as "passband-like", where the response remains fairly constant. Most analytically designed filters follow the sentiment, where the intention is to preserve low

[2]Our quick implementation divides classes by the first principle component of 5×5 image patches. PSNR is obtained by comparing an image to the interpolated image from a downsampled version of the original.

[3]It is unnecessary to zero-pad the spatial domain. We do so only for aesthetic purposes.



(a) Meshgrid of Padded Responses



(b) Top-down View of Typical Frequency Response

Figure 7.4: Magnitude Responses for Four Classes, $u = 2$.

frequencies as the uncorrupted remains of the original signal. The empirical design of our filter in Fig. 7.4 justifies the idea. Towards the edge of the filter at $\frac{\pi}{2}$, we can observe some humps just prior to the stop-band, which could conceivably represent the filter's attempt to push existing bandlimited frequencies outwards. The corners of the humps at $\left(\pm\frac{\pi}{2}, \pm\frac{\pi}{2}\right)$ end in a conic protrusion, where the MMSE's rendition of training pairs attempts to sharpen horizontal and vertical edges. Observing real-world image phenomena, the corner behavior tends to make sense, e.g. orientation of edges in buildings, objects, and characters and letters in any language. Finally, at high frequencies, $H(z_x, z_y)$ has near-zero values to suppress noise and aliasing.

We could, then, create a single filter with the common magnitude response seen in all class-specific filters. To accommodate different orientations, such an encompassing filter would almost certainly be zero-phase, and of course, given the symmetry of Fig. 7.4, real. The assertion is that we can maintain near the performance of [9], sacrificing little in terms of visual quality by the empirical aspect of our design. Concurrently, we could achieve the complexity of bicubic interpolation levels by avoiding the burden of classification.

There are many low-complexity, zero/linear-phase filters for image interpolation, and a majority of existing algorithms are in the form of M-band filters. Parks-McClellan filter design uses the Remez exchange algorithm, see Algorithm 5.1 in [32], to minimize ripple in both pass and stop-bands for flatness. M-band filters in image interpolation also require that all bands must satisfy

$$\sum_m H(ze^{-j2\pi k/M}) = 1. \tag{7.12}$$

Extensions of M-band filtering to two-dimensions [130, 91] scale axes independently (i.e. they interpolate vertically and then subsequently, horizontally), but retain the same thought of ensuring flatness in pass and stop-bands. The flatness specification as an essential design criterion suggests a philosophy that concentrates on suppression rather than enhancement. While excellent for purposes of reducing aliasing, high-frequency noise, and artifacts, we can, at best, expect image attributes at low-resolution to be left alone rather than elucidated. The case for M-band filters is further weakened because we can infer from Fig. 7.4 that the relationship between low and high-resolution image content requires some sharpening (for all edges, especially horizontal and vertical ones).

The proposed algorithm approaches filter design from an empirical standpoint by minimizing the MSE, similar to [9]. Rather than multiple filters, our approach differs with a single generic, zero-phase filter derived from a training set. As such, we simultaneously address complexity issues along with aforementioned issues plaguing M-band filters. What follows are several derivations of zero-phase MMSE filters that double image resolution (although u can be arbitrarily selected with minimal changes necessary.) The first two approaches impose zero-phase directly in the training. The second two approaches assume that, for some reason or another, we cannot or do not wish to re-train the data.

7.3.1 Pretraining Constraints on $H(z_x, z_y)$

Let us assume that we have a training set where a high-resolution image sample, vectorized as $\mathbf{y}_i$, is downsampled by two in each direction following an anti-aliasing filter to get the low-resolution sample, vectorized as $\mathbf{x}_i$. Recall from Sec. 7.2 that:

$$\mathbf{y} = A\mathbf{x} + \beta \tag{7.13}$$

where A can be reorganized into $h[n_x, n_y]$, which is $N \times N = 10 \times 10$, the desired impulse response. Without regenerating a new training set, we can directly enforce constraints, pre-training, to obtain zero-phase in the problem formulation on $H(z_x, z_y)$ or on its polyphase components, $E_m(z_x, z_y)$ for $m \in [0, 3]$. The underlying idea is to force symmetry on the impulse response about an axis, which can vary depending on the desired phase. Consequently, adding a zero-phase constraint effectively cuts the degrees of freedom in half.

In two dimensional representation, the first quadrant will be identical to a twice-flipped (horizontal and vertical) version of the third quadrant, where the pivot is $\left(\frac{N-1}{2}, \frac{N-1}{2}\right)$. The same goes for the second and fourth quadrant. Our pivot point is between samples since $u = 2$ dictates N even, meaning our filter is *Type II*. Should the upscaling factor be odd, we would have a *Type I* filter. Given 100 total coefficients, we need only find 50 because:

$$h[n_x, n_y] = h[N - n_x - 1, N - n_y - 1]. \tag{7.14}$$

In addition to (7.14), we must also satisfy one additional constraint that we have previously discussed in the context of M-band filtering and is common in wavelet theory [111]. To avoid regular gridding artifacts induced by unequal energies in the final image, we must have at least one zero at $H(\pm\pi, \pm\pi)$ and $H(\mp\pi, \pm\pi)$, all four π corners of our frequency response. The implication is that:

$$\begin{aligned} H(\pi, \pi) &= \sum_{n_x, n_y} h[n_x, n_y] e^{-j\pi(n_x+n_y)} = 0 \\ \leftrightarrow \quad \sum_{\text{even } n_x \pm n_y} h[n_x, n_y] &= \sum_{\text{odd } n_x \pm n_y} h[n_x, n_y]. \end{aligned} \tag{7.15}$$

The constraint in (7.15) is true of polyphase components for most image filters, and we can extend the property as the two-dimensional equivalent of (7.12).

There are a number of ways to simultaneously implement (7.14) and (7.15), and the next subsection describes one intuitive method.

7.3.2 Pretraining Constraints on $E_m(z_x, z_y)$

In its present form, A in (7.13) is not altogether conducive to the elimination of variables that (7.14) calls for. Additionally, (7.14) ignores the inherent polyphase structure that would clearly provide an implementation advantage in an MMSE framework that deals with A instead of H.

Our association of each row of A as the polyphase components, $E_m(z_x, z_y)$, of $H(z_x, z_y)$ defines four linear systems, which according to (7.14) are algebraically dependent. Let A_m, again, be the m^{th} row of A defined in (7.16).

$$A_m = \text{vectorize}\left\{E_m(z_x, z_y)\right\}, \qquad m = 0, 1, 2, 3. \tag{7.16}$$

$H(z_x, z_y)$ is a Type II filter, so the impulse response of the m^{th} polyphase component is *not* symmetric. Instead, the m^{th} polyphase component, denoted by $e_m(n_x, n_y)$, is equal to its diagonal counterpart polyphase component when horizontally and vertically flipped. Mathematically, this is,

$$\begin{aligned} e_0[n_x, n_y] &= e_3[N - n_x - 1, N - n_y - 1] \\ e_1[n_x, n_y] &= e_2[N - n_x - 1, N - n_y - 1]. \end{aligned} \tag{7.17}$$

If we define the "vectorize" operation as the concatenation of columns in an image patch into a one long column (MATLAB's `im2col` does the same thing), then

$$\begin{aligned} A_0[n] &= A_3[d^2-n-1] \\ A_1[n] &= A_2[d^2-n-1]. \end{aligned} \tag{7.18}$$

where d is the same d as Sec. 7.1, $d=\frac{N}{2}$.

We can use (7.18) to our advantage and define $\hat{A}$ with only two rows such that

$$\hat{A}=\begin{bmatrix} -A_0- \\ -A_1- \end{bmatrix} \tag{7.19}$$

where we wish to learn $\hat{A}$ in (7.19), which is half the size of the original A matrix. Define a "flip" operator that reverses the order of elements inside a vector. Then, the training pairs associated with the new $\hat{A}$ are derived from the original training pair, $(\mathbf{x}_i,\mathbf{y}_i)$, to give

$$\begin{aligned} \hat{\mathbf{x}}_i &= \begin{bmatrix} \mathbf{x}_i & \mathbf{x}+\mathrm{flip}\{\mathbf{x}_i\} \end{bmatrix} \\ \hat{\mathbf{y}}_i &= \begin{bmatrix} \hat{y}_i[0] \\ \hat{y}_i[1] \end{bmatrix} = \begin{bmatrix} y_i[0] & y_i[0]+y_i[3] \\ y_i[1] & y_i[1]+y_i[2] \end{bmatrix}. \end{aligned} \tag{7.20}$$

Here, $\hat{\mathbf{x}}_i$ has an extra column supplementing $\mathbf{x}_i$ that enforces symmetry, and $\hat{\mathbf{y}}_i$ is half the height of $\mathbf{y}_i$, also with an additional column. We wish to solve the new system of equations:

$$\hat{\mathbf{y}}=\hat{A}\hat{\mathbf{x}}+\hat{\beta} \tag{7.21}$$

After ensuring zero-phase in training through (7.21), we must now concentrate on the condition described in (7.15). Being that N is even and we are solving for a Type II linear phase filter, equal energy in polyphase components that are diagonal to each other is automatically ensured through (7.20). We can verify in terms of E_m that the upper left and its symmetrical lower right polyphase components have equal energy as do the upper right and its symmetrical lower left polyphase components. Therefore, we need only focus on polyphase components that are off-diagonal relative to one another.

The summation to the left of the equal sign in (7.15) contains $h[n_x,n_y]$ terms where n_x and n_y are either both even or both odd. The summation to the right of the equal sign in (7.15) contains $h[n_x, n_y]$ terms where only one of n_x and n_y is odd and the other is even. The positioning difference between the former and latter summations is, in fact, off-diagonal, and so (7.15) supplemented by (7.20) is, indeed, saying that the energy of any polyphase component is equal to the energy of the corresponding off-diagonal component.

Without (7.15), solving (7.21) is a simple MMSE solution. The solution considering (7.15) is similar, with an additional term. Let us denote $\hat{Y}$ as a $u \times T$ matrix and $\hat{X}$ as a $d^2 \times T$ (where T is the number of training pairs) of all normalized training points.[4] The optimization problem is written in (7.22)

$$\begin{aligned} \min_{\hat{A}} \quad & \left\| \hat{\mathbf{y}} - (\hat{A}\hat{\mathbf{x}} + \hat{\beta}) \right\|^2 \\ \text{s.t.} \quad & \hat{A}\mathbf{1} = \mathbf{1}, \end{aligned} \tag{7.22}$$

where $\mathbf{1}$ denotes a vector of ones of appropriate length.[5] We have written the constraint described by (7.15) as $\hat{A}\mathbf{1} = \mathbf{1}$ to regularize the energy. The problem is convex, and we can determine the closed form by writing the Lagrangian and taking its derivative. We have solved for the d-dimensional Lagrange optimization variable shown in (7.23).

$$\boldsymbol{\lambda} = \frac{R_{xy}R_{xx}^{-1}\mathbf{1} - \mathbf{1}}{\mathbf{1}^T R_{xx}^{-1}\mathbf{1}}. \tag{7.23}$$

The final solution is given as:

$$\hat{A} = R_{xy}R_{xx}^{-1} - \mathbf{1}^T R_{xx}^{-1} diag(\boldsymbol{\lambda}), \tag{7.24}$$

where $diag(\boldsymbol{\lambda})$ is a square matrix with the entries of λ on the diagonal and zeros elsewhere.

7.3.3 Post-Training Symmetry Enforcement

If the filters have already been found via Sec. 7.1, we may not wish to re-train the data. One plausible situation for not re-training may be that the training set is not be available. Another scenario may be that we do not wish to put in the

[4]We can eliminate the need to solve for β by incorporating it into X and Y.

[5]In $A\mathbf{1} = 1$, the vector $\mathbf{1}$ left of the equal sign is of size $d^2 \times 1$. The vector $\mathbf{1}$ to the right of the equal sign is of size $\frac{u^2}{2} \times 1$.

computation time to re-train all points in Ω to find an MMSE filter that satisfies (7.21).

In response, a simple approach would be to take an existing filter, ensure that the magnitude response looks like Fig. 7.4 (a good majority of classes, if not all of them, exhibit the response), impose zero-phase on the filter coefficients by retaining some coefficients and copying others, and lastly, scale each polyphase component to sum to one. We can impose zero-phase with (7.14) or (7.17).

As one might have guessed, the heuristic nature of the approach leads to several design freedoms that undoubtedly yield different solutions. For example, which polyphase component should we preserve with (7.17), and which polyphase component can we replace? There are equally numerous comparison metrics that favor one set of parameters over others.

We have determined the final coefficients through visual inspection. Yet, in the end, as long as we have a magnitude response resembling Fig. 7.4, we have found that visual quality does not vary too much by altering our other degrees of freedom.

7.3.4 Post-Training Frequency Domain Phase Elimination

In future sections, we explore arbitrary scaling factors given a fixed training set. In such cases, there must be a commonality tying the filter design together. Using the magnitude responses seen in Fig. 7.4 as the common factor, it is only natural to impose zero-phase through the frequency domain.

Without any zero-padding, we take the $N \times N$ two dimensional DFT, $H(k_x, k_y)$, of an existing filter, $h[n_x, n_y]$, that has been generated via methods in Sec. 7.1. This can be the same filter that we have chosen to modify in Sec. 7.3.3. Next, we let the magnitude response be the desired full response, which effectively discards the original phase of $H(k_x, k_y)$. That is,

$$\hat{H}(k_x, k_y) = |H(k_x, k_y)|$$

Since h real $\leftrightarrow$ H symmetric, $\hat{h}$ is real as well. Couple this with $\angle \hat{H}$ equaling zero, and $\hat{h}$ becomes a real, symmetric, zero-phase filter based with the same magnitude response as h. Hence, by letting the DFT of $\hat{H}$ be entirely real, the mesh of $|\hat{H}|$ reflects that of Fig. 7.4 with none of the phase of H.

Because the DFT is a uniformly sampled version of the Discrete Time Fourier Transform (DTFT), there are some implementation issues associated with what type of filter an inverse DFT can provide. The DFT frequencies are limited to discrete values of $\frac{2\pi k}{N}$ ranging from $[0, \frac{2\pi N-1}{N}]$. Therefore, even filter lengths will still generate Type-I-like symmetry, offsetting with a shift of $z_x^{-\frac{1}{2}} z_y^{-\frac{1}{2}}$.

One remedy is to train odd filters lengths so that the axis of symmetry is located on samples at integer locations. Doing so would defeat the purpose of not re-training, however. Another suggestion is to alter the filter support without perturbing the magnitude response. We can do so with border replication or zero-padding. This course of action may affect the performance by introducing artifacts since the h on which $\hat{h}$ had been built was originally intended for 10×10 inputs. Moving to an 11×11 filter would involve pixel values that haven't been considered during training, and it is unclear how the change in dimensions alters other parameters and the final result.

In the end, though, numerical problems caused by the DFT are very minor. The filter $\hat{h}$ is still linear phase, and the added shift only serves to shift the entire image, which we can rectify after we are finished interpolating.

7.4 Arbitrary Scaling Factors Using Frequency Domain Responses

Standalone learning algorithms like [9], by design, can be rather inflexible. Because the purpose is to learn the natural relationship between input/output training pairs of fixed pixel dimensions, inferences can be made only of that particular relationship. Unforeseen adjustments such as added noise models require alterations to the training set, without which place generalization beyond the intended scope of the original algorithm. For interpolation, we expect data from a $d \times d$ image patch to resolve a single pixel to $u \times u$ pixels. Any change to the problem statement, i.e. resolving a single pixel to $v \times v$ pixels, where $u \neq v$, may exceed the intended purpose of the algorithm. Consequently, to change u, the upsampling factor, the current implementation of [9] must obtain an entirely new training set.

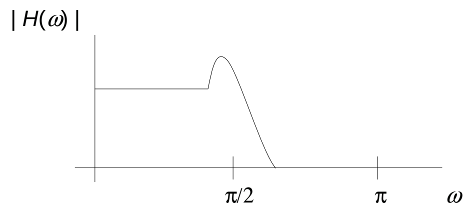
At present, common and successful learning algorithms have mostly considered spatial domain representations. This has evolved with the good reasons described in Chapter 5. Besides the costly process of re-training the algorithm, another option is to broaden the versatility by increasing robustness with various pre-processing techniques. Pre-processing alone, though, does not drive at the root of the problem and is thus limited in the extent of its generalization.

Under the assumptions of Sec. 7.3,[6] we do, nevertheless, have knowledge of the non-zero portion of the filter magnitude response for any scaling factor of u on the interval $\omega_u = [-(\frac{\pi}{u} + \delta_u), \frac{\pi}{u} + \delta_u]$. By establishing constraints from Sec. 7.3.1 through Sec. 7.3.4, we have focused the learning aspect of the proposed algorithm on the magnitude response seen in Fig. 7.4. We propose to develop spatial domain filters using knowledge of the frequency response to learn the relationship between low and arbitrarily high resolutions. We can do so by our assumptions of zero-phase and also mapping the non-zero band of $H(\omega)$ in $\omega_2 = [-(\frac{\pi}{2} + \delta_2), \frac{\pi}{2} + \delta_2]$ to a new interpolating filter with transfer function of $H_u(\omega)$ with a corresponding non-zero band in $\omega_u = [-(\frac{\pi}{u} + \delta_u), \frac{\pi}{u} + \delta_u]$. We also stipulate a cutoff band for the remainder of the frequency response outside ω_u; the magnitude response of $H_u(z)$ here should be near zero.

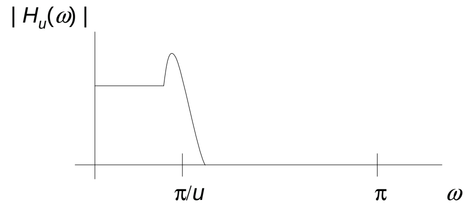
There are two steps in our proposed rational scaling algorithm. The first step is to obtain the appropriate $h_u[n]$ given the $2\times$ interpolating filter from Sec. 7.3. We would like to obtain the response in Fig. 7.5(b) from the response of Fig. 7.5(a). For visualization purposes, let Fig. 7.5(a) be the one-dimensional representation of the two-dimensional magnitude response shown in Fig. 7.4.

Besides finding the spatial domain filter $h_u[n]$ to determine pixel *values*, the second step finds the grid to place the pixels in the correct *location*. Placement of pixel values is simple for the integer-based interpolation scenario: upsample

[6]Extensions to 2-D can be easily derived, and we work exclusively with 1-D in this section. Throughout this section, $h[n]$, the impulse response of $H(\omega)$ is the 1-D version of the filter proposed in Sec. 7.3.



(a) Interpolation by 2× (Trained Response)



(b) Interpolation by $u\times$

Figure 7.5: Given Magnitude Responses for Interpolation Factors of 2 and u

the low-resolution image by u and then filter using $h_u[n]$. Placement of pixel values for the non-integer-based interpolation scenario requires that one have filter taps at non-integer locations or some other special resampling technique. The two subsections that follow are devoted to the topics of integer and non-integer scaling factors. In both subsections, the number of calculation to upsample and produce filters is minimal, and the proposed algorithms are capable of producing of interpolation kernels on-the-fly.

7.4.1 Interpolation by Integer Factors

The phase of an image is arguably its defining characteristic. Zero-phase filters are especially attractive in that they will not add, subtract, or otherwise alter the original phase of an image. The contention is that image phase does not significantly vary from resolution to resolution, and the proposed algorithm, as a zero-phase filter, fits especially well. We intend to enhance existing attributes without altering content defining information, so we preserve the empirical design aspect by using the MMSE zero-phase filter $h[n]$ from calculations in Sec. 7.3 as a starting point. The goal, then, is to find $h_u[n]$ that corresponds to $h[n]$, only the superresolution factor will be $u = 2M$ instead of two. Discussions that follow in this section involve integer values of u.

Given $H(z)$, the question is, how shall we fill in the remaining high-frequency bands of $H_u(\omega)$ that have no corresponding value in $H(\omega)$? Researching the issue, we have collected a diverse assortment of solutions. One interesting idea would be to zero-pad to the right of the passband of Fig. 7.5(a) to fill in Fig. 7.5(b). Video postprocessing techniques almost universally advise against zero padding a filter response in the frequency domain. On the whole, we concur that induced ringing artifacts by a synthetically zeroed response may be noticeable, albeit our experimentation suggests that the effect is not severe.

A more reserved suggestion is to interpolate the interpolating filter itself. Interpolated finite impulse response (IFIR) filters [82] are a simple way to regularize an upscaled filter to find $H_u(z)$. The principle behind IFIR filters for our purposes is to shrink the non-zero portion of the frequency response to its appropriate size while zeroing out higher cutoff frequencies. Generating IFIR filters is shown in the general block diagram of Fig. 7.6.

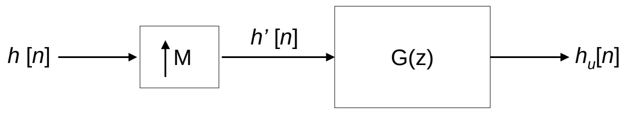
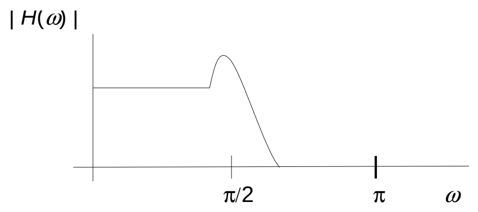
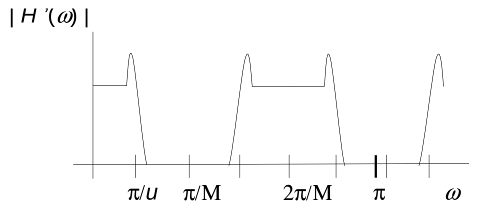


Figure 7.6: IFIR Block Diagram

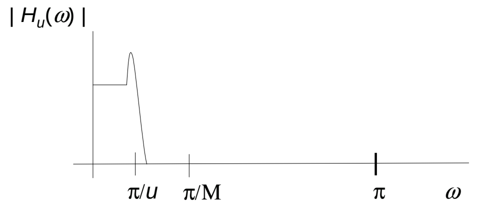
Upsampling $H(z)$ by M for $H'(z) = H(z^M)$ creates periodicity in the frequency domain at every $\frac{2\pi}{M}$. In the interval $[-\pi, \pi]$, the copy that we are most interested in is the one from $-(\pi/u + \delta)$ to $\pi/u + \delta$, the centered frequency, making $G(z)$ a lowpass filter. According to [82], we can use any type of filter for $G(z)$. Many references suggest equiripple filters from the Parks-McClellan algorithm because of its sharp transition at the cutoff frequency, a well-sustained passband, and good point-wise approximation of zero in the stopband. Another choice is the maximally flat (Daubechies) filter because of its many zeros at π. The entire process for the development of $H_u(z)$ can be easily followed in Fig. 7.6.



(a) Trained FIR Filter Response of **h**



(b) Upsampled FIR Filter Response of **h'**



(c) Desired IFIR Filter Response of $\mathbf{h}_u$

Figure 7.7: Frequency Responses for Each Stage of Cascade in Fig. 7.6

Integer scaling is convenient in that samples after upsampling are regularly located on the upper left-hand corner locations in the high-resolution grid. After finding $h_u[n_x, n_y]$, the task is trivial. One only has to filter the entire image and $y_u[n_x, n_y] = h_u[n_x, n_y] * x[n_x, n_y]$ is the final, high-resolution image.

7.4.2 Interpolation by Rational Scaling Factors

Multirate approaches to non-integer interpolation factors are difficult because straightforward upsampling by $u = 2M/L$, where L does not divide $2M$, means that some low-resolution pixel values cannot be placed on the high-resolution grid. Because we filter after upsampling, the task is difficult because the coefficients of $h_u[n_x, n_y]$ utilize the pieces of information that have been displaced onto non-integer locations.

Before proceeding, we clarify some possible confusion on the difference between the interpolation factors M and u. M is the interpolation factor of the *filter*, while u is the interpolation factor of the *signal*. Because the original filter $h[n_x, n_y]$ already interpolates by two in each direction, to generate $h_u[n_x, n_y]$, we need only interpolate by $M = \frac{u}{2}$. Not coincidentally, u is also the patch dimension after superresolving.

Literature on rational scaling factors is by no means sparse, many of the works stemming from a multiresolution filterbanks perspective [130, 91]. The common view of rational scaling factors can be implemented in Fig. 7.8, where the user wishes to interpolate by a rational factor u/L. The first portion of Fig. 7.8 is identical to the integer scaling case. Only a L-downsampler has been added to the end of the entire procedure, and the u-upsampler has been purposely placed first. After applying the filter $H_u(z)$, there should be no aliasing when we downsample by L provided that $L < u$ and the signal is somewhat bandlimited.

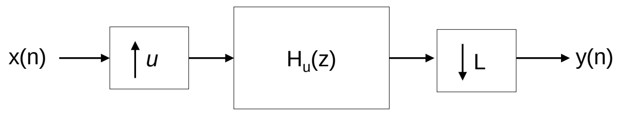


Figure 7.8: Typical Rational Scaling Framework

Computing every detail in Fig. 7.8 is almost certainly very computationally complex, especially for close ratios of u and L (e.g., scaling by 1.1 requires $u/L = 11/10$.) Conventional approaches, most notably those concentrating on multiframe superresolution [96], use concepts of non-uniform sampling and exclusively place the responsibility of generating values onto the high-resolution grid in the co-domain. Because correlation matrices do not exist, they rely on assumed point spread functions and spectral density estimates. Cross-correlation approaches are analytically derived, and thus very intuitive and logical in what they set out to achieve. The analytical nature of such models and the assumptions of common blur functions require that algorithms make a case of relevance to the problem at hand. The relevancy point is moot in learning-based solutions because they rely on empirical data, but the idea of filtering on the high-resolution grid can certainly be adopted in our case.

Fig. 7.9 shows the super-high-resolution grid in the example of an interpolation by 3/2. Here, $u = 3$ and $L = 2$. The grid depicts the resolution after upsampling by u and not the end high-resolution grid, which is $\left(\frac{1}{L}\right)^{th}$ of the size. Each box denotes a pixel location. The dark boxes are the locations with the original pixel values; the lightly shaded boxes are the final locations in the final image after downsampling by L; the split boxes are the locations where the original pixel values and the values to be generated overlap; and the unfilled boxes are the locations that we do not care about. Again, we generate $H_u(z_x, z_y)$, but now we need only calculate pixels at the lightly shaded boxes and filter for $\left(\frac{1}{L}\right)^2 = \frac{1}{4}$ of the total number of pixels. Moreover, we need only process the polyphase components corresponding to the shaded boxes.

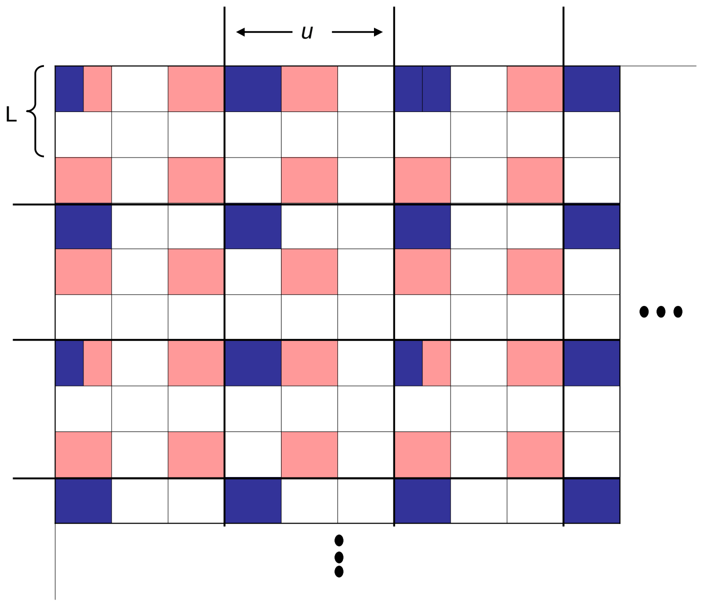


Figure 7.9: Super-high resolution grid, $u\times$ the original image size.

Conceptually, the final algorithm resembles a filter that has non-integer offsets, where instead of filter taps of $h[n_x, n_y]$ located at $n_{x,y} = 1, 2, 3, \cdots$, offsets $n_{x,y}$ can take on values like 1.3 or 3.2, etc. Such intuition can be very powerful, and ultimately, we maintain the bottom line that the computation is still directly proportionate to the end scaling factor of u/L.

7.5 Results and Analysis

Several approaches, all of which are shown in Fig. 7.10, were proposed in Sec. 7.3 with the same objective of attaining zero-phase given a magnitude response. The IFIR filter rational scaling approach of Sec. 7.4 can employ all subsections of Sec. 7.3, but for continuity, we use Sec. 7.3.3.[7] The training set, Ω, consists of preprocessed (mean-shifted, variance normalized, and vectorized) low-resolution/high-resolution pairs $(\mathbf{x}_i, \mathbf{y}_i)$. Unlike [9], we do not apply pre-sharpening to the high-resolution training samples, which gives better numerical accuracy. When measuring the PSNR values, we downsample by a fixed factor with MATLAB's `imresize` command (MATLAB applies an anti-aliasing filter that defaults to 11×11 taps), upsample by the same factor, apply the proposed filter $H(z_x, z_y)$, and measure the differences between the original image and the interpolated one. We numerically and visually compare to various methods that have been cited throughout the body of this work, and we benchmark using MATLAB's `profile` function.

Our goal remains achieving "near"-optimal visual results, with considerable reductions in computation time. There are some MATLAB efficiency issues when measuring complexity that might slightly skew results. Nonetheless, the order of magnitude improvement in computation time is significantly noticeable. Average function call times for various interpolation methods are shown in Table 7.1. The proposed methods in Sec. 7.3 require nothing more than single-pass linear filtering, and process an entire image in only a few milliseconds. The number of operations are on the order of bicubic interpolation. Other algorithms of note include a 20-tap halfband filter, a first-vertical and then horizontal effort culminating in an overall two-pass filtering system, which apparently doubles the filtering time (probably due to implementation issues in MATLAB) despite having one fifth of the filter taps as

[7]The briefly mentioned zero-padding approach of Sec. 7.4 for rational scaling can only employ the frequency domain scheme of Sec. 7.3.4, which ironically is not quite zero-phase.

the proposed method. We have chosen 20 taps for $2\times$ interpolation (rational scales require more or fewer taps) because it is difficult to enforce the energy requisite (that the sum of all taps in a single polyphase filter component must equal one) with fewer than 10 total filter taps. As seen in Table 7.1 and discussed in Sec. 7.3, [12] requires a formidable number of calculations. Depending on the total number of classes, C, runtimes will vary proportionately to the sum of

1. Computations the posterior probability C values
2. $d \times d$ filtering for C individual class
3. The final summing of the filtered responses

for *every* low-resolution pixel, which does not begin to address memory accesses and space issues. Runtimes in Table 7.1 also include minor issues such as vectorizing every input $\mathbf{x}$.

Table 7.1: Computational Complexity Benchmarks in $2\times$ Interpolation

Time (seconds) to Interpolate Various Sequences

Sequence	RS [12]	TBRS [11]	NEDI [74]	Halfband	Sec. 7.3.3
Barbara	846.48	45.09	72.96	0.087	0.042
Bus	182.54	21.93	15.61	0.022	0.0101
Paris	182.61	21.62	15.41	0.022	0.0095
City	182.99	21.78	15.67	0.022	0.0096

The differences in implementations from Fig. 7.10(d), Fig. 7.10(e), and Fig. 7.10(f), subsections of Sec. 7.3, are manifested in overall luminance levels, small numerical differences, and some very slight visual defects in select areas of Fig. 7.10(e) and Fig. 7.10(f) that are not quite as apparent in Fig. 7.10(d). For example, there appears to be a halo effect along the bus's top edge in Fig. 7.10(e) and Fig. 7.10(f) that is not present in Fig. 7.10(d). The selective discomfitures are logical as Sec. 7.3.3 and Sec. 7.3.4 enforce zero-phase more heuristically than Sec. 7.3.2. Rather than starting somewhere in the middle of the solution to avoid training (and depending on how the original magnitude response is chosen, possibly using information from a single class), Sec. 7.3.2 begins at the root of training to find the most accurate filter possible.

For rational interpolation factors, M-band filters in Fig. 7.10(c) inevitably lead to slightly blurry results. Sharper images over a larger spatial grid necessarily demand new and higher frequencies. M-band filters cut off or zero-out edge-defining frequencies by degrading the passband before $\frac{\pi}{M}$ instead of pushing the existing bandwidth outwards. Hence, the low-pass information appears averaged across a larger area, much like the outcome of bicubic interpolation. The "humps" in the proposed magnitude response enhance those higher frequencies that M-band filters are designed to suppress in an intelligent way, which avoids the problem of blurring content.

Visual results generated by our implementation of [12] perform surprisingly worse than [11]. One likely explanation stems from the fact that [12] linearly averages outputs through a mixture of experts framework. While effective for reductions in MSE, summing outputs from individual linear interpolators, not all of which necessarily reflect the true values, does not quite make visual sense. Weighting by posterior probabilities should theoretically lessen the impact of the averaging, but the underlying principle is flawed when, already, linear interpolation tends to cause the observed blurring and washed-out effect, and averaging a group of linear interpolators exacerbates the problem. In contrast, [11] and the proposed algorithm choose a single filter for a specific result.

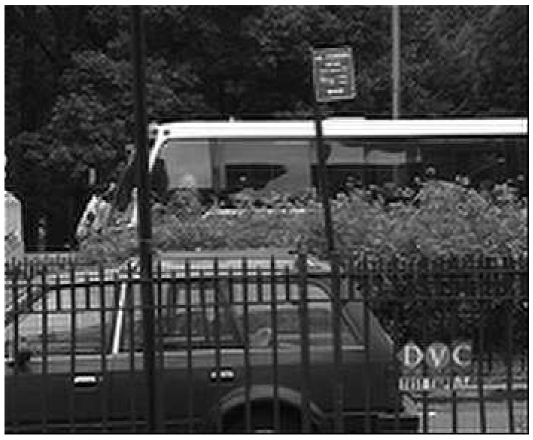
The argument for classification-based image interpolation on the whole is weakened when $|\Omega|$ is small, e.g. 6×10^6 training points, and the proposed, single filter produces sharper images. Furthermore, [12] relies on a presharpening step, which puts the correctness of the image construction/reconstruction in question. That said, when $|\Omega|$ is large (i.e., a large training set), both [12] and [11] perform very well, and numerical differences and visual acuity, as expected, outcompete the proposed method. The choice of algorithm depends on the circumstance and

(a) Original Image

(b) Bicubic Interpolation

(c) M^{th}-Band Polyphase Interpolation Filter

(d) Methods Described by Sec. 7.3.1 and Sec. 7.3.2


(e) Method Described by Sec. 7.3.3

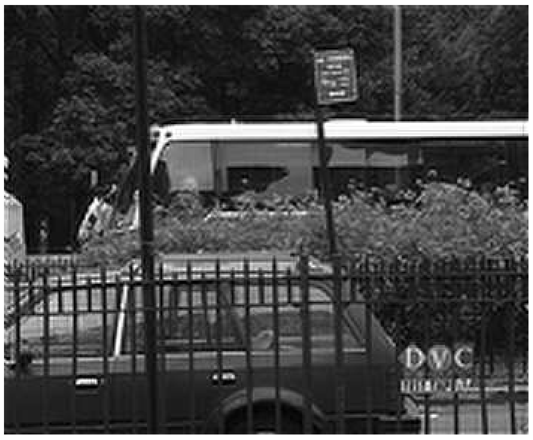
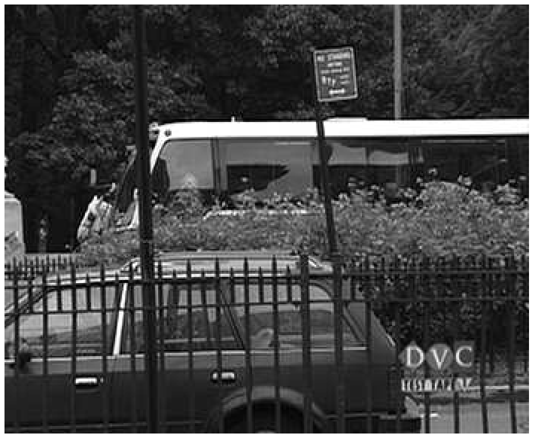
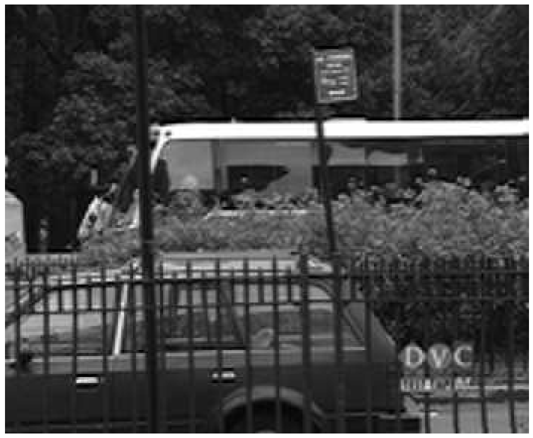

(f) Method Described by Sec. 7.3.4

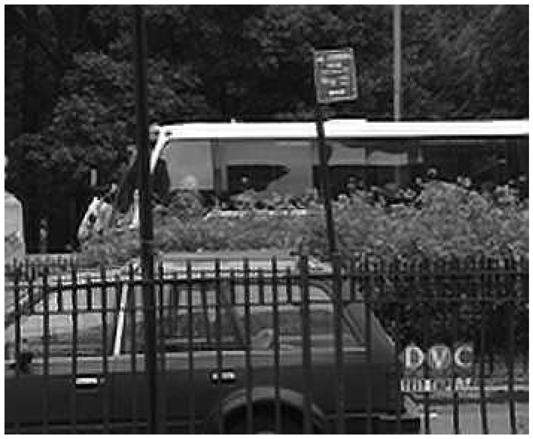
Figure 7.10: Interpolation of Bus Sequence Using Various Methods



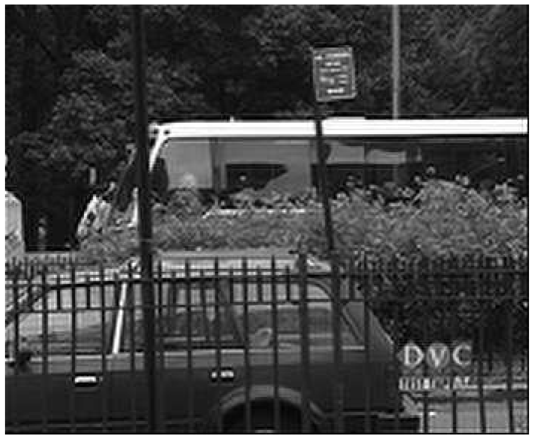
(a) Original Image



(b) 100-Class Resolution Synthesis [12] (Pre-sharpening Filters Implemented)



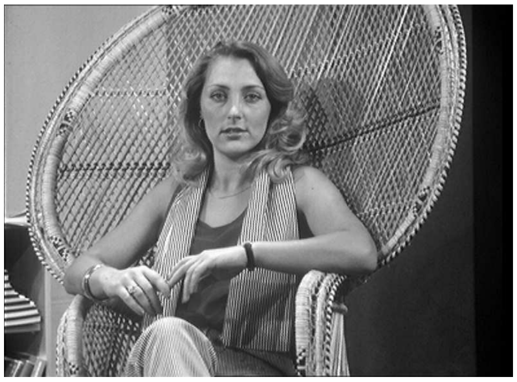
(c) 128-Class Tree-Based Resolution Synthesis [11]



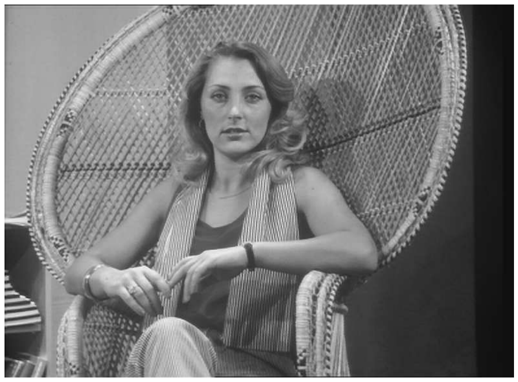
(d) Method Proposed by Sec. 7.3.3

Figure 7.11: Comparisons to Various Classification-Based Filtering Algorithms

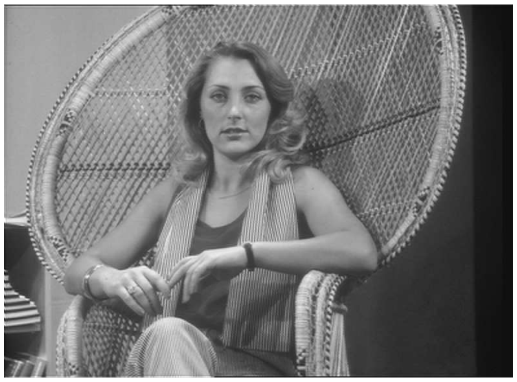
(a) Original Image



(b) 100-Class Resolution Synthesis [12]



(c) 128-Class Tree Based Resolution Synthesis [11]



(d) Method Proposed by Sec. 7.3.3

Figure 7.12: Comparisons to Various Classification-Based Filtering Algorithms

the trade off of accuracy for speed.

The PSNR values are given in Table 7.2. Interestingly enough, the proposed method unexpectedly outperforms most of the algorithms in every experiment. The pleasant surprise may have something to do with recent studies that low-resolution and high-resolution patch-based manifolds are not close to being isometric [112]. Hence, the built-in specificity of classification algorithms poorly represent high-resolution numerical results. Also surprising is the poor performance of the halfband filter. If we ignore the anti-aliasing filter applied prior to downsampling in the setup described in the first paragraph of the results section, we would boost the numbers a bit. We do not do so because the scenario is unrealistic as most of the time, we wish to superresolve un-aliased images. Of course, PSNR and MSE are well-known to be poor metrics of visual quality, but they do, nonetheless attest to the "accuracy" of the proposed algorithm.

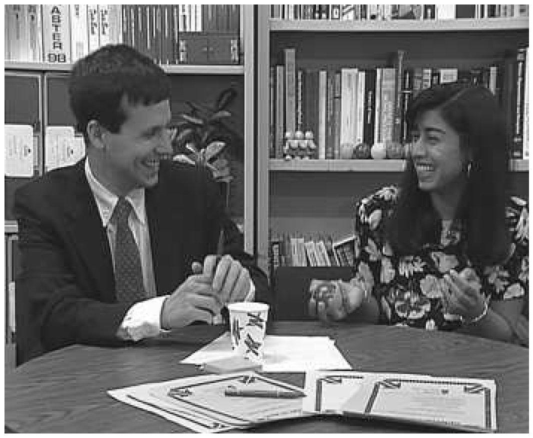
Table 7.2: PSNR Results for 2× Interpolation

PSNR (dB) of Interpolation for Various Sequences

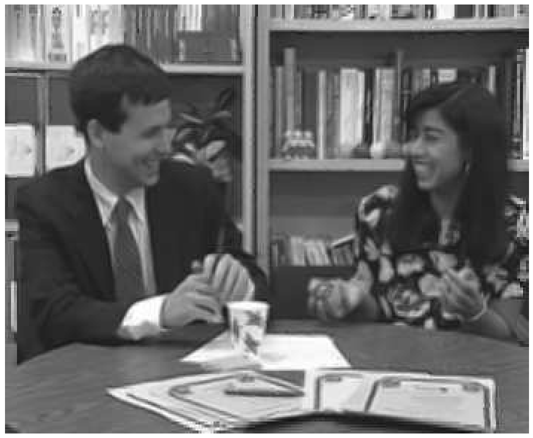
Sequence	RS [12]	TBRS [11]	Bicubic	Halfband	Sec. 7.3.3
Barbara	28.30	26.56	28.64	26.44	28.32
Bus	23.78	24.26	24.80	23.00	24.72
Paris	21.40	21.62	22.06	20.22	21.54
City	27.70	27.40	27.46	25.04	26.90

Also shown are edge-directed interpolation algorithms [74, 62] in Fig. 7.13. Subpixel Edge Localization [62] produces a somewhat cartoonish image, and Li's edge-directed interpolation algorithm relies on a covariance matrix that is inherently low-resolution. Differences between the edge-enhancing algorithms and the proposed algorithm are especially emphasized in the face of the man and woman, where one can hardly call the image generated by [74] and [62] high-resolution.

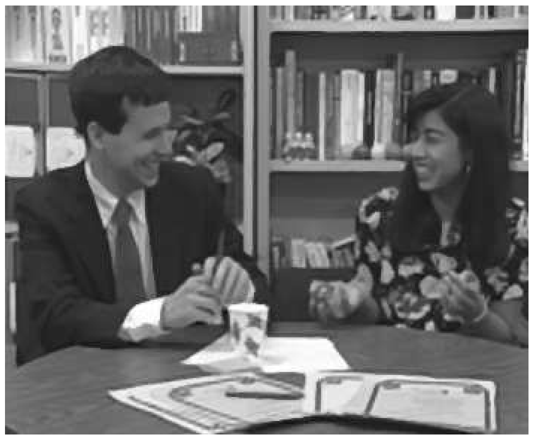
Finally, specific testing of rational scaling factors can be seen in Fig. 7.14



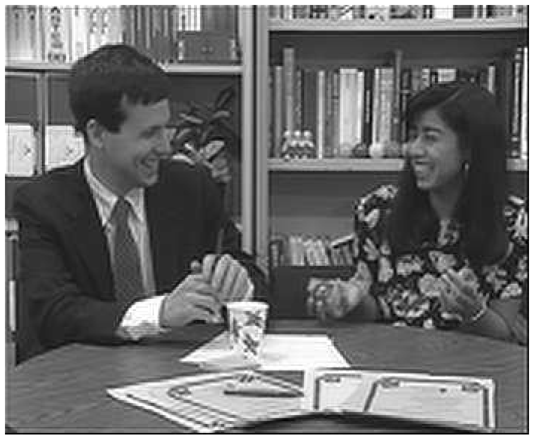
(a) Original Image



(b) Edge Directed Interpolation [74]



(c) Subpixel Edge Localization Interpolation [62]



(d) Method Proposed by Sec. 7.3.3

Figure 7.13: Comparisons to Edge-Directed Interpolation Algorithms

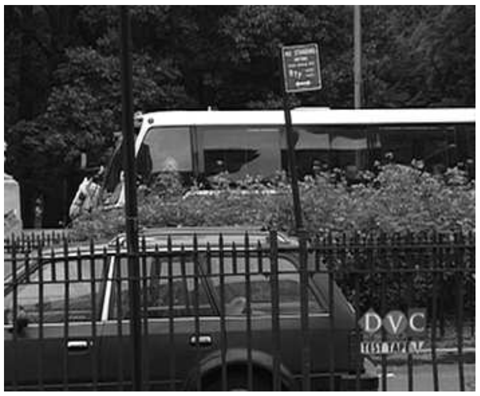
where the scaling factor is 4/3. One should be careful to set $G(z)$ from Sec. 7.4 to have a passband that includes all of the non-zero portions of Fig. 7.4. This is simple if we recall that the cutoff frequency is $\omega_c = \pm\left(\frac{\pi}{u} + \delta_u\right)$. We have chosen $\delta_u = 0.2/u$. To generate flat pass and stopbands, we set the initial number of taps in $G(z)$ to 20 and then scaled linearly according to $M = \frac{u}{2}$. Not discussed in Sec. 7.4 are some necessary energy regularizations to avoid regular mismatches in the polyphase components of the image, which we have heuristically rectified by normalizing all polyphase filter components. The mismatches probably come about through the use of Parks-McClellan filter design theory in $G(z)$. We can easily address this by using wavelet-type constraints in filter design, where all the energy in the subbands must be equal. Also, for a scaling factor of 4/3 apparently the IFIR filter has become non-zero-phase. This fact does not affect the results too much as the filter is still near linear phase.

Evident from Fig. 7.14, the rational scaling of Sec. 7.4 gives very well-defined edges in Fig. 7.14(e). Texture has also been preserved quite well, if one notices the trees, bushes, and horse head. We have included results from zero-padding the magnitude response, briefly mentioned in Sec. 7.4, in Fig. 7.14(d). The rippling artifacts are actually fairly unnoticeable, probably due to a small window size. Overall, the zero-padded results are comparable in performance with the proposed IFIR algorithm. Both algorithms have better edge-resolution than the M-band and bicubic interpolator.

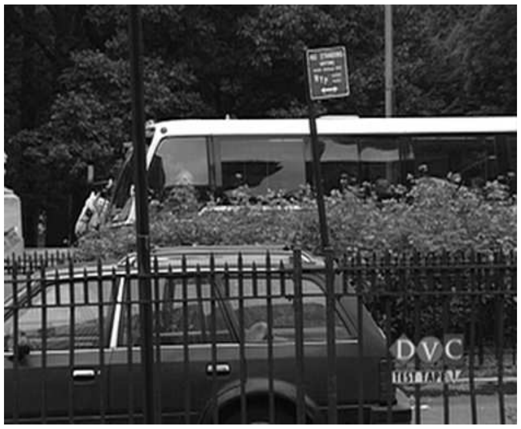
7.6 Summary

We have reviewed classification-based interpolation, which has traditionally been developed in the machine-learning setting. We have introduced a new perspective by analyzing class-based filter design as a composition of polyphase components. We have proposed a zero-phase, MMSE interpolation filter based on a training set. Finally, we have proposed an approach to extend the MMSE zero-phase interpolation algorithm to arbitrary scaling factors.

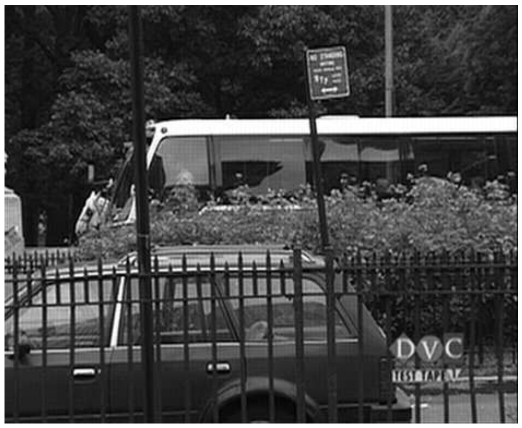
To design the zero-phase filter, we have proposed four different methods of



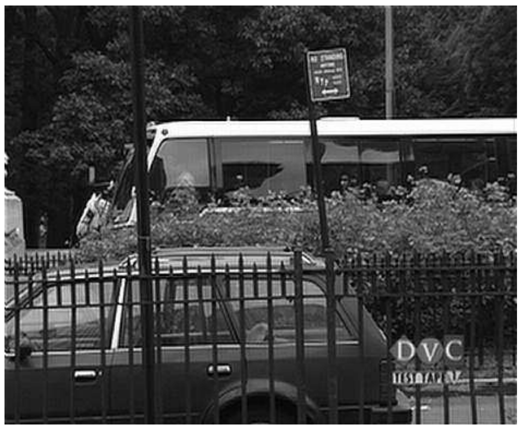
(a) Original Image



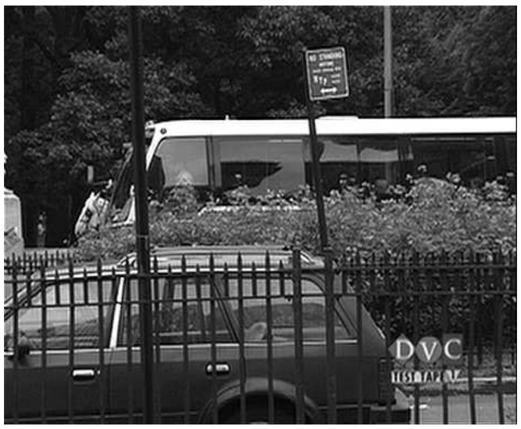
(b) Bicubic Interpolation



(c) M-Band Filtering



(d) Magnitude Zero-Padding



(e) IFIR Filter

Figure 7.14: 4/3 Rational Scaling of the Bus Sequence

obtaining such a zero-phase filter:

- Train real and symmetric coefficients in $H(z_x, z_y)$
- Train real and symmetric coefficients in $E_m(z_x, z_y)$
- Eliminate phase in the spatial domain
- Eliminate phase in the frequency domain

The second methodology is a more specific version of the first. Both of the first two items in the above list are implemented prior to training. Therefore, one would need to retrain in order to determine the correct filter coefficients. The third and fourth items in the list are implemented after one has already trained for the algorithm in [9]. Using a single class or a combination of a few classes, one ignores several portions of the filter to make the coefficients symmetric about the center axis.

Although the fourth item is not quite symmetric, it is used for arbitrary interpolation factors, both integer and rational. By using IFIR filters or zero-padding, we can find the correct filter responses for any scaling factor. To scale by non-integer, rational factors, we ignore unnecessary operations to keep complexity low.

The complexity of the algorithm is on the order of bicubic interpolation. Depending on the training set, the visual quality is comparable to higher-complexity algorithms, including the original classification-based interpolation approaches but at a fraction of the computational cost. The proposed algorithm produces better results than edge-directed interpolation approaches.

7.7 Acknowledgements

The text of Chapter 7 is adapted from *A Zero-phase Filter for Image Interpolation*, Karl Ni and Truong Nguyen, May 2008, manuscript submitted to the *IEEE Transactions on Image Processing.* The dissertation author was the primary

author of this publication, and the listed co-author directed and supervised the research that forms the basis for this chapter.

8 Conclusions and Future Work

The road leading to a goal does not separate you from
the destination; it is essentially a part of it.

—Charles de Lint

We have focused on the analysis of patch-based algorithms for image interpolation. Several interpolative techniques have been introduced. The work done in this thesis may be summed up as follows:

Image Distribution Model A probability distribution function has been introduced for image patches. The underlying framework assumes a GMM with a super-Gaussian around $\mathbf{0}$. Conceptual and intuitive descriptions of what the mixtures and center peak represent have been described.

Adaptable k-Nearest Neighbor Interpolation Two dimensional image filters have good resiliency when they are built for the general case. There is an optimal k given a training set and test point for any image patch. The optimal k value is determined by how many training points are close in distance. Small k denotes an image patch that is well-represented by the training set. Large k denotes an image patch that is not.

Support Vector Regression Interpolation Optimal kernels can be learned with a sufficient set of training points in the support vector regression problem. Training points can also be weighted depending on their relevancy to the regression.

An Optimal Zero-phase Filter for Interpolation MMSE filtering under several classes is often unnecessary. Often, we can be approximate the filters with a single filter to yield fairly good results. Using polyphase filters, we can choose arbitrary interpolation factors to resize the image.

Despite the thorough investigation of the image patch domain as well as discriminant and nonparametric methods, we have opened several avenues for future design and analysis. Each path of exploration has the potential to lead to a grand, unifying methodology in the application of learning to image interpolation, and more generally image processing.

The behavior of the PDF of image patches under varying amounts of training still requires further research. The proposed model works well enough (or at least better than currently accepted techniques), but is by no means exact. For example, in smaller collections of training images, the center peak in the distribution becomes more and more sharp, approaching a delta function. Moreover, as mentioned in Chapter 3, the statistical significance of the metrics used to measure the goodness of fit of the multivariate distributions has not been fully investigated. The depth of research required is considerable, and would involve some statistical mathematics that are beyond the scope of this thesis. There are several references that address the issue of statistical significance [36] and multivariate analysis [100] using tables of significance levels to associate with goodness of fit metrics, but none exist for the multivariate Laplace distribution case, and there is no unifying metric to analyze our center curve. With tables of statistical significance levels, it becomes possible to know when to use the model, and when the goodness of fit levels require another model.

A more immediate need, the solution from which several other image processing and computer vision problems may benefit, is finding the replacement for the Euclidean distance. [104] deals directly with this problem for image patches in one of its chapters. Because k-NN and classification algorithms rely on searches for points with the closest distances, many alternative distance metrics usually concentrate on complexity reduction and quality of search. The manifestation is some sort of feature space representation [84] or dimensionality reduction/manifold learning. Concepts such as semi-definite embedding [128] are especially attractive in the latter of the two.

Other topics that warrant future research include the choice of kernel function. Our algorithms have exclusively used Gaussian (RBF) kernels with varying covariance matrices, number of features, and overall σ parameters. For Chapter 5, because DCT coefficients are known to Laplace distributed (though recent studies [6] have suggested the distribution is closer to a Cauchy distribution), a kernel choices that reflect the corresponding distribution may yield better results. One can also address the further generalization of the currently-implemented linear combination of kernels to a convex function of kernels. Relevant papers involve hyperkernels [90].

In the SVR chapters, Chapters 5 and 6, prediction is carried out on four targets independently (for $2\times$ superresolution). Because interrelationships in pixel values and DCT coefficients are prevalent and the values themselves are highly correlated, integrating the concept of dependency is only logical. One approach includes learning inter-pixel relationships with nonlinear predictors (perhaps kernel methods) while supplementing the original task of learning pixel values. The inclusion of image autocorrelation in the feature space $\mathcal{F}$ may be an alternate route. Other recent developments with potential in this respect concern the learning of vector valued functions using operator-valued kernels [78] or multi-regression SVR [95], broached in Sec. 5.3. By associating a measure of smoothness in [78] or a cost function in [95] to model relationships among $y_1, y_2, \ldots y_U$ of $\mathbf{y}$, vector valued functions predict individual components simultaneously, and further exploration of the topic could benefit pixel prediction by doing so jointly.

Finally, video-processing-based algorithms are always concerned with the robustness of proposed algorithms to noise. Currently, the proposed learning algorithms actually enhance noise due to windowed and non-globalized approaches. This is a problem that learning-based methods in general face. The relation to be learned usually depends on the training set that is assumed. With a training set that does not contain any noise, the proposed algorithm knows only to replicate high-resolution data from uncorrupted low-resolution data. Possible solutions to the problem almost certainly include some type of preprocessing of input vectors. There may also be room for improvement by researching how to adjust parameters in the presence of noise. Exploration of the topic would not only benefit the current framework, but most learning algorithms as well.

Bibliography

[1] *The MOSEK optimization toolbox for MATLAB manual. Version 4.0 (Revision 16)*, 2006.

[2] J. I. Agbinya. Interpolation using the discrete cosine transform. *Electronic Letters*, 28(20), 1992.

[3] V. R. Algazi, G. E. Ford, and R. Potharlanka. Directional interpolation of images based on visual properties and rank order filtering. volume 4, page 30053008, 1991.

[4] Z. Alkachouh and M. G. Bellanger. Fast dct-based spatial domain interpolation of blocks in images. *IEEE Transactions on Image Processing*, 9(4):729–732, 2000.

[5] J. Allebach and P. W. Wong. Edge-directed interpolation. In *IEEE International Conference on Image Processing*, 1996.

[6] Y. Altunbasak and N. Kamaci. An analysis of the dct coefficient distribution with the h.264 video coder. In *IEEE International Conference on Acoustics Speech and Signal Processing*, volume 3, pages 177–180, 2004.

[7] A. F. Atiya. Estimating the Posterior Probabilities Using the K-Nearest Neighbor Rule. *Neural Comp.*, 17(3):731–740, 2005.

[8] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–74, 1997.

[9] C. B. Atkins and C. Bouman. *Classification based methods in optimal image interpolation*. PhD thesis, Purdue University, 1998.

[10] C. B. Atkins, C. Bouman, and J. Allebach. Tree-based resolution synthesis. In *Proceedings of the Conference on Image Processing, Image Quality, Image Capture Systems*, pages 405–410, 1999.

[11] C. B. Atkins, C. A. Bouman, and J. P. Allebach. Tree-based resolution synthesis. In *PICS*, pages 405–410, 1999.

[12] C. B. Atkins, C. A. Bouman, and J. P. Allenbach. Optimal image scaling using pixel classification. *IEEE International Conference on Image Processing*, 3:864–867, 2001.

[13] F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 33–40, New York, NY, USA, 2005. ACM Press.

[14] C. Bohm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[15] M. B. G. Bonnerot and M. Coudreuse. Digital filtering by polyphase network: Application to sample rate alteration and filter banks. *IEEE Transactions on Acoustics, Speech, and Signal Procecssing*, 24:109–114, April 1976.

[16] S. Borman and R. Stevenson. Simultaneous multi-frame MAP super-resolution video enhancement using spatio-temporal priors. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 469–473, Kobe, Japan, Oct. 1999.

[17] S. Borman and R. Stevenson. Simultaneous multi-frame MAP super-resolution video enhancement using spatio-temporal priors. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 469–473, Kobe, Japan, Oct. 1999.

[18] C. A. Bouman. Cluster: An unsupervised algorithm for modeling Gaussian mixtures. Available from http://www.ece.purdue.edu/~bouman, April 1997.

[19] S. Boyd and L. Vandenberghe. *Convex Optimization.* USA Cambridge University Press, 2004.

[20] R. N. Bracewell, K. Y. Chang, A. K. Jha, and Y. H. Wang. Affine theorem for two-dimensional fourier transform. *Electronic Letters*, 29(3), 1993.

[21] A. J. Broder. Strategies for efficient incremental nearest neighbor search. *Pattern Recognition*, 23(1-2):171–178, 1990.

[22] D. Burr, M. Morrone, and D. Spinelli. Evidence for edge and bar detectors in human vision. *Vision Research*, 4:419–431, 1989.

[23] CalPhotos. *Cal Photos Image Collections.*

[24] F. M. Candocia and J. C. Principe. Super-resolution of images based on local correlations. *IEEE Transactions on Neural Networks*, 10(2):372, March 1999.

[25] W. K. Carey, D. B. Chuang, and S. S. Hemami. Regularity-preserving image interpolation. In *International Conference on Image Processing*, pages 901–908, 1997.

[26] W. K. Carey, D. B. Chuang, and S. S. Hemami. Regularity-preserving image interpolation. *IEEE Transactions on Image Processing*, 8(9):1293–1297, September 1999.

[27] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76(1):1–12, January 2008.

[28] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. *IEEE Conference on Computer Vision and Patter Recognition*, 01:275–282, 2004.

[29] P. Clark and F. Evans. Distance to nearest neighbor as a measure of spatial relationships in populations. *Ecology*, 35(4):445, 1954.

[30] P. Clark and F. Evans. Generalization of a nearest neighbor measure of dispersion for use in k dimensions. *Ecology*, 60(2):316, 1979.

[31] W. S. Cleveland and S. J. Delvin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of American Statistical Association*, 83(403):596–610, 1988.

[32] D. S. P. Committee. *Programs for Digital Signal Processing.* IEEE Press, Piscataway, NJ, USA, 1979.

[33] A. G. Constantinides and R. A. Valenzuela. A new recursive interpolator with applications in transmultiplexer design. In *International Conference on Signal Processing*, 1981.

[34] T. Cover. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-14(1), January 1968.

[35] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[36] R. D'Agostino and M. Stephens. *Goodness of Fit Techniques*, volume 68 of *Statistics, textbooks and monographs.* Marcel Dekker, Inc., 1986.

[37] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. Technical report.

[38] C. A. Dávila and B. R. Hunt. Training of a Neural Network for Image Superresolution Based on a Nonlinear Interpolative Vector Quantizer. *Applied Optics*, 39:3473–3485, July 2000.

[39] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistics Society*, 39:1–38, 1977.

[40] L. P. Devroye, L. Gyorfi, A. Krzyzak, and G. Lugosi. On the strong universal consistency of nearest neighbor regression function estimates. *The Annals of Statistics*, 22(3):1371–1385, 1994.

[41] L. P. Devroye and T. J. Wagner. Distribution-free consistency results in nonparametric discrimination and regression function estimation. *The Annals of Statistics*, 8(2):231–239, 1980.

[42] W. D'haes, D. V. Dyck, and X. Rodet. An efficient branch and bound search algorithm for computing k nearest neighbors in a multidimensional vector space, 2002.

[43] L. Diao, K. Hu, Y. Lu, and C. Shi. A method to boost support vector machines. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 463–468, 2002.

[44] P. Diggle. On parameter estimation and goodness-of-fit testing for spatial point patterns. *Biometrics*, 35(87), 1979.

[45] R. Duda, P. Hart, and D. Stork. *Pattern Classification.* Wiley Interscience, 2nd edition, 2000.

[46] O. Ekiz and A. G. Constantinides. Phase linearity in polyphase filters. In *ICASSP '00: Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference*, pages 321–323, Washington, DC, USA, 2000. IEEE Computer Society.

[47] T. Eltoft, T. Kim, and T. Lee. On the multivariate laplace distribution. *IEEE Signal Processing Letters*, 13(5), May 2006.

[48] C. Fraley and A. E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[49] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, March/April 2002.

[50] K. S. Fu. *Digital Pattern Recognition.* Springer-Verlag, 2nd edition, 1980.

[51] P. M. Goebel and A. N. Belbachir. Single image superresolution interpolation by wavelet support vector regression. *Wavelets and Applications Semester and Conference*, 2006.

[52] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[53] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[54] M. Grant, S. Boyd, and Y. Ye. *CVX: Matlab Software for Disciplined Convex Programming.*

[55] J. A. Hartigan. *Clustering Algorithms.* John Wiley and Sons, Inc., 1975.

[56] S.-H. Hong, R.-H. Park, S. Yang, and J.-Y. Kim. Image interpolation using interpolative classified vector quantization. *Image Vision Comput.*, 26(2):228–239, 2008.

[57] H. Hou and H. Andrews. Cubic splines for image interpolation and digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26, 1978.

[58] J. Huang and D. Mumford. Statistics of natural images and models. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 1999.

[59] P. Indyk. Nearest neighbors in high-dimensional spaces, 2003.

[60] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. of 30th STOC*, pages 604–613, 1998.

[61] R. Jacobs and M. I. Jordan. Adaptive mixture of experts. *Neural Computation*, 3:79–87, 1991.

[62] K. Jensen and D. Anastassiou. Subpixel edge localization and the interpolation of still images. *IEEE Transactions on Image Processing*, 4:285–295, 1995.

[63] C.-M. Kao, X. Pan, M. Anastasio, and P. L. Riviere. An interpolation method using signal recovery and discrete fourier transform. *IEEE Medical Imaging Conference*, 1998.

[64] B. S. Kim and S. B. Park. A fast k nearest neighbor finding algorithm based on the ordered partition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):761–766, 1986.

[65] K. I. Kim, M. Franz, and B. Scholkopf. Kernel hebbian algorithm for single-frame super-resolution. *Statistical Learning in Computer Vision*, pages 135–149, 2004.

[66] V. Kober, M. A. Unser, and L. P. Yaroslavsky. Spline and sinc signal interpolations in image geometrical transforms. In N. A. Kuznetsov and V. A. Soifer, editors, *Proc. SPIE Vol. 2363, p. 152-161, 5th International Workshop on Digital Image Processing and Computer Graphics (DIP-94), Nikolai A. Kuznetsov; Viktor A. Soifer; Eds.*, volume 2363 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 152–161, Jan. 1995.

[67] S. Kotz, T. J. Kozubowski, and K. Podgorski. *The Laplace Distribution and Generalizations: A Revisit with Applications to Communications, Economics, Engineering, and Finance.* Birkhauser, 2001.

[68] B. Kullis, M. Sustik, and I. Dhillon. Learning low-rank kernel matrices. *International Conference on Machine Learning*, 2006.

[69] J. T.-Y. Kwok. Support vector mixture for classification and regression problems. In *Proceedings of the Fourteenth International Conference on Pattern Recognition*, volume 1, pages 255–258, Brisbane, Qld., Australia, 1998.

[70] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[71] A. Lee, K. Pedersen, and D. Mumford. The complex statistics of high-contrast patches in natural images, 2001. private correspondence.

[72] A. B. Lee, K. S. Pedersen, and D. Mumford. The nonlinear statistics of high-contrast patches in natural images. *Int. J. Comput. Vision*, 54(1-3):83–103, 2003.

[73] M. Li and T. Nguyen. Discontinuity-adaptive de-interlacing scheme using markov random field model. October 2006.

[74] X. Li and M. Orchard. New edge-directed interpolation. *IEEE Transactions on Image Processing*, 10:1521–1527, 2001.

[75] C. A. Lima, A. L. Coelho, and F. J. V. Zuben. Hybridizing mixtures of experts with support vector machines: Investigation into nonlinear dynamic systems identification. *Information Sciences*, 177:2049–2074, 2007.

[76] D. C. Lin and P. M. Chau. Objective human visual system based videoquality assessment metric for low bit-rate video communication systems. August 2006.

[77] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7), July 1989.

[78] C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17, 2005.

[79] C. Miravet and F. B. Rodriguez. A hybrid mlp-pnn architecture for fast image superresolution. *Lecture Notes Computer Science*, 2714:401, 2003.

[80] I. Narsky. Goodness of fit: What do we really want to know? In *PHYSTAT2003*, pages 70–74, 2003.

[81] A. Netravali and B. Hasskell. *Digital Pictures: Representation, Compression and Standards*. New York Plenum Press, 2nd edition, 1995.

[82] Y. Neuvo, D. Cheng-Yu, and S. Mitra. Interpolated finite impulse response filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(3), June 1984.

[83] N. Nguyen, P. Milanfar, and G. Golub. A computationally efficient superresolution image reconstruction algorithm. *IEEE Transactions on Image Processing*, 10:573–583, 2001.

[84] K. Ni, S. Kumar, and T. Q. Nguyen. Learning the kernel matrix for superresolution. *IEEE Conference on Multimedia Signal Processing*, August 2006.

[85] K. Ni, S. Kumar, T. Q. Nguyen, and N. Vasconcelos. Single image superresolution based on support vector regression. *International Conference on Acoustics, Speech, and Signal Processing*, May 2006.

[86] K. Ni and T. Q. Nguyen. Kernel resolution synthesis for superresolution. *International Conference on Acoustics, Speech, and Signal Processing*, 2007.

[87] K. Ni and T. Q. Nguyen. Color image superresolution. *International Conference on Image Processing*, To appear in September 2007.

[88] K. S. Ni and T. Q. Nguyen. Image superresolution using support vector regression. *IEEE Transactions on Image Processing*, to appear.

[89] H. Niemann. An efficient branch-and-bound nearest neighbour classifier. *Pattern Recogn. Lett.*, 7(2):67–72, 1988.

[90] C. S. Ong and A. J. Smola. Machine learning using hyperkernels. In *International Conference on Machine Learning*, 2003.

[91] S. Oraintara and T. Q. Nguyen. Image/video scaling algorithm based on multirate signal processing. In *ICIP (2)*, pages 732–736, 1998.

[92] M. Palaniswami and A. Shilton. Adaptive support vector machines for regression. In *Proceedings of the 9th International Conference on Neural Information Processing*, 2002.

[93] D. Pavlov, D. Chudova, and P. Smyth. Scaling-up support vector machines using boosting algorithm. In *International Conference on Pattern Recognition*, 2000.

[94] K. S. Pedersen. *Statistics of Natural Image Geometry.* PhD thesis, University of Copenhagen, Denmark, 2003.

[95] F. Perez-Cruz, G. Camps-Valls, E. Soria-Olivas, J. J. Perez-Ruixo, A. R. Figueiras-Vidal, and A. Artes-Rodriguez. Multi-dimensional function approximation and regression estimation. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 757–762, London, UK, 2002. Springer-Verlag.

[96] R. Prendergast and T. Q. Nguyen. Digital video super-resolution. *Submitted to IEEE Transactions on Image Processing*, 2007.

[97] S. Qiu and T. Lane. Multiple kernel learning for support vector regression. Technical report, University of New Mexico, 2005.

[98] D. Rajan and S. Chaudhuri. An mrf-based approach to generation of super-resolution images from blurred observations. *J. Math. Imaging Vis.*, 16(1):5–15, 2002.

[99] K. R. Rao and P. Yip. *Discrete Cosine Transforms: Algorithms, Advantages, Applications.* Academic Press, Inc., 1990.

[100] A. Rencher. *Methods of Multivariate Analysis.* Wiley Series in Probability and Statistics. Wiley Interscience, second edition, 2002.

[101] P. J. L. Riviere and X. Pan. Mathematical equivalence of zero-padding interpolation and circular sampling theorem interpolation with implications for direct fourier image reconstruction. *SPIE Medical Imaging Conference*, 1998.

[102] A. Roorda. Human visual system - image formation. *The Encyclopedia of Imaging Science and Technology*, 1:539–557, 2002.

[103] B. Scholkopf. Statistical learning and kernel methods. Technical report, Microsoft Research Limited, 2000.

[104] G. Shakhnarovich. *Learning Task-Specific Similarity.* PhD thesis, Massachusetts Institute of Technology, 2005.

[105] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice.* The MIT Press, 2005.

[106] K. Shi and S. Zhu. Mapping natural image patches by explicit and implicit manifolds. In *CVPR07*, pages 1–7, 2007.

[107] T. Sikora. Mpeg digital video–coding standards, 1997.

[108] V. N. Smelyansky, P. Cheeseman, D. A. Maluf, and R. D. Morris. Bayesian super-resolved surface reconstruction from images. In *Computer Vision and Pattern Recognition*, pages 1375–1382, 2000.

[109] M. Smid. Closest-point problems in computational geometry. *Handbook of Computational Geometry*, 2000.

[110] D. Squire and T. Pun. A comparison of human and machine assessments of image similarity for the organization of image databases, 1997.

[111] G. Strang and T. Nguyen. *Wavelets and Filterbanks.* Wellesley-Cambridge Press, 1996.

[112] K. Su, Q. Tian, Q. Que, N. Sebe, and J. Ma. Neighborhood issue in single-frame image superresolution. IEEE Computer Society, 2005.

[113] S. Szedmak, J. Shawe-Taylor, and E. Parado-Hernandez. Learning via linear operators: Maximum margin regression. Technical report, PASCAL, Southampton, UK, 2005.

[114] J. Taguchi, K. Kido, and K. Sano. Directional image filter respectively adjusted to edge and flat regions. *Systems and Computers in Japan*, 30(8):72–78, 1999.

[115] A. Temizel and T. Vlachos. Image resolution upscaling in the wavelet domain using directional cycle spinning. *Journal of Electronic Imaging*, 14(4):040501, 2005.

[116] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.

[117] P. Tsai and T. Archarya. Image up-sampling using discrete wavelet transform. In *Joint Conference on Information Services*, Advances in Intelligent Systems Research, 2006.

[118] L. Tuncel. On the slater condition for the SDP relaxations of nonconvex sets. *Operations Research Letters*, 29:181–186, 2001.

[119] M. Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38, November 1999. IEEE Signal Processing Society's 2000 magazine award.

[120] G. Uytterhoeven. *Wavelets: Software and Applications.* PhD thesis, Katholieke Universiteit Leuven, 1999.

[121] P. P. Vaidyanathan. Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proc. of IEEE*, 78(1):56–93, Jan. 1990.

[122] P. P. Vaidyanathan. *Multirate Systems and Filter Banks.* Prentice Hall, Upper Saddle River, NJ, USA, 1993.

[123] P. Vandewalle, S. Susstrunk, and M. Vetterli. Superresolution images reconstructed from aliased images. In *Proc. SPIE/IS&T Visual Communications and Image Processing Conference*, volume 5150, pages 1398–1405, 2003.

[124] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[125] E. Vazquez and E. Walter. Multi-output support vector regression. 13^{th} *IFAC Symposium on System Identification*, 2003.

[126] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[127] R. Vollgraf, M. Scholz, A. Meinertzhagen, and K. Obermeyer. Nonlinear filtering of electron micrographs by means of support vector regression. *Lecture Notes Computer Science*, 2714:401, 2003.

[128] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *Int. J. Comput. Vision*, 70(1):77–90, 2006.

[129] S. Yang and K. Hong. Bilateral interpolation filters for image size conversion. In *IEEE International Conference on Image Processing*, pages 986–989, 2005.

[130] S. Yang and T. Q. Nguyen. Interpolated mth band filters for image size conversion. In *ICIP (3)*, pages 891–894, 2001.

[131] Z. Zhang, D.-Y. Yeung, and J. T. Kwok. Bayesian inference for transductive learning of kernel matrix using the tanner-wong data augmentation algorithm. *Proceedings of the* 21^{st} *International Conference on Machine Learning*, 2004.