METHODS AND APPLICATIONS OF IRREGULAR SAMPLING AND SCATTERED DATA INTERPOLATION OF DIGITAL IMAGES

REZA SHAHIDI

Your file Votre référence
ISBN: 0-612-89669-2
Our file Notre référence
ISBN: 0-612-89669-2

METHODS AND APPLICATIONS OF IRREGULAR SAMPLING AND SCATTERED DATA INTERPOLATION OF DIGITAL IMAGES

BY

REZA SHAHIDI

A Thesis submitted to the

School of Graduate Studies

in partial fulfillment of the

requirements for the degree of

Master of Engineering

FACULTY OF ENGINEERING AND APPLIED SCIENCE

MEMORIAL UNIVERSITY OF NEWFOUNDLAND

2003

Abstract

This thesis examines the problem of the irregular sampling of images and the scattered data interpolation or reconstruction of images from these irregular samples. Since in our tests only a small number of samples are taken, the entire process can be viewed as low bitrate image compression. We look at two different existing irregular sampling algorithms, Farthest Point Sampling (FPS) and skewness-based sampling. We then propose two new progressive irregular sampling algorithms, *gaps* and Faster Farthest Point Sampling (FFPS). FFPS is, as its name suggests, quicker than FPS, while *gaps* addresses concerns regarding the quality of reconstructions from irregular samples.

Note that reconstruction is the flip side of sampling. One existing fast reconstruction algorithm called Multilevel B-Spline Approximation does not work well on irregular sampling techniques which take samples relatively far away from edges, so a new modification of this algorithm, called New Edge-Directed Multilevel B-Spline Approximation (NEDMBA), which uses image inpainting, is introduced.

Finally, we apply Faster Farthest Point Sampling to digital image halftoning to

create a new algorithm called Farthest Point Halftoning. This new dither array generation algorithm is compared to other existing standards, for example the Modified Blue Noise Mask and the Void and Cluster method, and is shown to perform favorably.

Acknowledgments

First, I would like to thank my supervisor Dr. Cecilia Moloney for her patience, guidance, and financial support. Her focus was invaluable in the completion of this thesis. I would also like to thank Dr. Mahmoud Haddara for suggesting that I pursue a M. Eng. degree at Memorial University and for his consistent and continuous interest in my research and general well-being. Dr. Giovanni Ramponi of the University of Trieste highlighted the link between irregular sampling and halftoning and also provided the `femme` image used for the test set. The help of Dr. Dennis Peters and his introduction to me of the C++ Standard Template Library was much appreciated. Finally, my parents both deserve much praise for their understanding and encouragement - I dedicate this thesis to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General

With the increasing digitization of image data in diverse fields, e.g. in aerial photography and medical imaging, the quantity of image data is becoming unmanageable. Although larger storage devices and higher-bandwidth transmission lines can provide a partial remedy to this problem, the sheer quantity of image data requires more compact representations of images. For example, images from digital mammography can occupy approximately 200 megabytes uncompressed per examination [1]. Thus it is easy to see that in just one hospital or medical centre, terabytes of digital image data for this application may be generated over a very short period of time. This, along with other similar examples, has provided the motivation for the use of image compression.

There are two types of image compression:

- Lossless compression: No information is lost between the original and compressed images. Examples are arithmetic coding, dictionary-based compression schemes, and lossless JPEG.

- Lossy compression: The original image can only be approximately obtained from the compressed form. Some information in the image cannot be reconstructed, but higher compression ratios may be achieved. Examples are fractal image compression and ordinary JPEG.

For many applications, lossy compression is acceptable as the difference between the original and compressed images is below a threshold of visibility or of detail loss. Lossy compression may become necessary if storage space or transmission rates are limited. For example, for SLIP and PPP Internet connections, images from the World Wide Web, though potentially very important in content, can be bottlenecks as they may take a long time to transmit. Transmission after lossless compression could still be too slow, so it may become incumbent to reduce bandwidth requirements by using lossy compression.

Low bitrate still-image compression can also play a major role in low bitrate video compression since some high compression image coding algorithms can be generalized from two to three dimensions, or can be combined with motion compensation for video coding (e.g. in MPEG-4). Some applications for which low bitrate image

and video compression (with high compression ratios) are necessary or advantageous are video-conferencing, cellular video-telephones, deep space communications and multimedia (for instance digital encyclopedias and electronic newspapers) [2, 3].

This thesis will focus on image coding with high compression ratios. The widely known compression standards from the Joint Photographic Experts Group (JPEG) are described in the following section in which we show why they are not necessarily suitable for low bitrate image compression.

This thesis will also look at digital image halftoning, the problem of rendering or printing a continuous-tone (contone) or many-level image with fewer levels, often only two (black and white). Halftoning is studied as a result of the particular approach that is taken in this thesis with respect to low bitrate image compression (irregular sampling/interpolation) part of which (irregular sampling) can be applied to the problem of halftoning.

1.2 Motivation

Though used widely as an image compression standard, the JPEG file format is not suited to low bitrates due to the presence of blocking artifacts and halos around edges (see Figure 1.1). JPEG is named after the acronym of its founding body, the Joint Photographic Experts Group. Blocking is a feature of most block-based image coding techniques, and though there have been many attempts, e.g. [4], to reduce

(a) Original cameraman

(b) Low bitrate cameraman JPEG

Figure 1.1: Example of blocking artifacts with low bitrate JPEG (app. 22:1 compression ratio)

the severity of this problem for JPEG, it is hard to completely eliminate. There is however a standard called progressive JPEG, also from the Joint Photographic Experts Group, which allows the progressive display of JPEG images with the same final image quality as non-progressive JPEG. The decoder for such images is currently implemented in most web browsers, but the blocking phenomenon still makes low-bitrate JPEG unattractive.

The Joint Photographics Expert Group also introduced JPEG 2000 which is based on wavelet technology. This was accepted as an international standard in December 2000 [5]. JPEG 2000 converts the blocking artifacts evident with JPEG to fine lines and gradation of intensity or color within each block [6]. JPEG 2000 also provides Region of Interest (ROI) coding and progressive transmission. The cost, however, is a slower and more complex implementation than the Discrete Cosine Transform algorithm used by conventional JPEG. It is therefore desirable to devise

new low bitrate image compression schemes which combine the high efficiency of the commonly used JPEG with the less noticable artifacts of the JPEG 2000 standard. The major low bitrate image compression schemes used in practice fall into three categories: waveform, second generation and fractal coding techniques [3]. Each of these classes of methods however creates various artifacts in the compressed image. Thus in this thesis a less explored technique for low bitrate image compression is turned to, namely a combination of irregular sampling followed by scattered data interpolation.

As mentioned in the previous section, this thesis also examines digital image halftoning, the reduction of color or gray level depth of an image. Halftoning has many applications, among them printing on bi-level printers, display on LCDs, the transmission of faxes, and image compression and transmission (where it is used in conjunction with inverse halftoning, the reverse process). Inverse halftoning can be achieved using various filtering approaches or neural networks, among other techniques, but these are beyond the scope of this thesis. In this thesis, a new halftoning algorithm is compared to the standard available methods.

1.3 Problem definition

This thesis sticks to gray-scale images for the sake of simplicity, both for low bitrate image compression and halftoning. As mentioned in the body of this thesis, however,

many algorithms herein can be extended to operate on color images as well, despite the fact that color images are inherently more complex due to the presence of three color planes that are not necessarily correlated. The main problem that is explored in this thesis is the one of lossy compression of grayscale images with the maximum possible compression ratio (minimum possible bitrate) for use in low bitrate image compression/transmission without the noticable artifacts found in many methods.

As will be seen in Chapter 6 of this thesis, irregular sampling can be used for the problem of halftoning, which has a rich and lengthy history. In the bilevel case, the visual illusion of different gray levels is created by using different configurations and/or sizes of black and white dots. From a close proximity, it is obvious that the image is composed of only two shades. From a normal viewing distance, however, the human visual system cannot discriminate between the dots and instead visually integrates over them, creating the sensation of gray tones. The goal is to create halftones with as few visible textures and patterns as possible in reasonably short amounts of time.

1.4 Approach to the Solution

One way to compress an image is to record only the intensities of some pixels of the image while completely eliminating the intensity data for others. The problem then becomes one of deciding which pixels to keep and which to throw away in order to

simultaneously minimize the amount of information lost and the number of pixels retained. Such a process is known as sampling.

One obvious method to sample an image is to do so regularly. For a one-dimensional signal, regular sampling means that samples are taken at regularly spaced intervals. This is the same for two or higher-dimensional signals, but in these cases, samples are chosen on a regular grid, with spacings set to possibly different constants in each dimension. For both the 1-D and higher dimensional cases, if the sampling frequency is too low (below the Nyquist frequency, which is twice the maximum frequency in the spectrum of the image), then aliasing occurs, meaning that there will be some frequency distortion in the reconstruction from the samples. A regularly sampled image at two different relatively low rates and their reconstructions using an algorithm called Multilevel B-Spline Approximation (MBA) [7] are shown in Figure 1.2. Clearly if the spacing is too large, information is lost, but as the spacing decreases, the reconstruction becomes more faithful to the original image. The presence of step patterns in the reconstructions is a result of this reconstruction algorithm acting on regular samples, but the main idea of aliasing comes across (frequency shifts are evident). The problem is with the sampling and not the MBA algorithm. Ideal reconstruction, though not implementable because of the infinite extent of the reconstruction filter, gives the same type of aliasing results.

One method to avoid the aliasing associated with regular sampling is the use of irregular sampling, or sampling which is not on a regular grid. As observed in the

(a) Original Lena image

(b) MBA reconstruction of Lena regularly subsampled with period 15 pixels

(c) MBA reconstruction of Lena regularly subsampled with period 4 pixels

Figure 1.2: Aliasing on reconstruction with regularly subsampled Lena image

paper by Robinson and Ren [8] the problem of choosing the best placement of samples for a given number of samples is ill-posed and NP-complete. Since the problem is NP-complete, the solution time is clearly super-polynomial. In fact, if n_s samples of an $m \times n$ pixel image are to be taken, an exhaustive search of the "best" (giving the reconstructions of highest quality) n_s locations requires $\binom{mn}{n_s}$ operations. So for images of typical dimensions, it becomes necessary to use strategies and heuristics to simplify the selection of a near-optimal solution.

Sometimes images and other signals are irregularly sampled not by choice but by necessity. For example, it may be impossible to obtain astronomical images regularly through time because of atmospheric conditions, like cloud or rain [9]. This is also the case with ecological and environmental data because of the high cost of regularly sampling large areas of land or water in space and time as well as other practical

difficulties [10]. Such applications are not investigated in this thesis, but provide areas of extension of the methods presented herein.

Once the irregular samples have been obtained, lossless compression can be performed and the compressed samples may be transmitted or written to a file. Subsequently, it becomes necessary to try to recover the original image from the sampled data with as high fidelity as possible to the original. This problem of recovery is referred to as scattered data interpolation, because in general the samples are not taken on a regular grid. Scattered data interpolation propagates the given information at the samples to a usually fine regular grid; in the context of an image, to all the pixels of the original image grid.

For further image compression, it is also possible to quantize the scattered data intensities either finely or coarsely, with the tradeoff of inferior reconstructed image quality. This can be viewed as lossy compression of the sample data.

Irregular sampling and scattered data interpolation are tightly coupled problems, so it is not unusual for a sampling algorithm to be optimized for a specific reconstruction algorithm or vice versa. The quality of reconstructions for a given algorithm is also important because it potentially means that fewer samples need to be taken, and this increases the compression ratio. The entire process is shown in flow diagram form in Figure 1.3, adapted from [11].

As previously mentioned, the problem of digital halftoning is also looked at in this thesis, as an application of irregular sampling. There are many different approaches

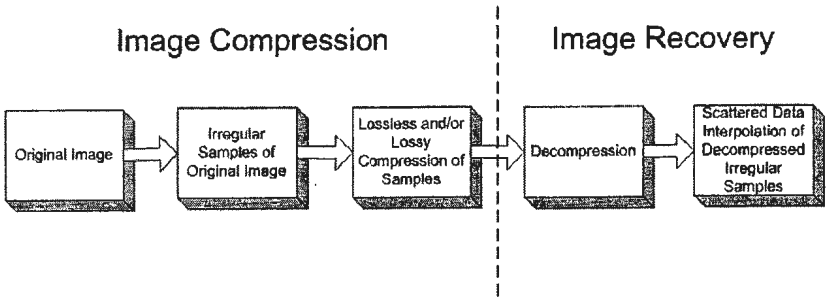


Figure 1.3: Overall data flow for image compression/decompression

to digital halftoning that appear in the literature, and these can be divided into two classes based on how the halftoning is performed [12]:

- Point processes: The image to be halftoned is thresholded with a fixed array of values point-by-point. A point process uses ordered dither if the formation of the threshold array is completely deterministic, and is otherwise known as stochastic screening. Ordered dither processes can be further subdivided into clustered-dot and dispersed-dot methods. Ordered dither and stochastic screening techniques are more efficient and easily parallelizable than neighborhood processes.

- Neighborhood processes: As the name suggests, not only a given pixel's intensity in an image to be halftoned is needed; those in its immediate vicinity are also utilized. This makes neighborhood processes more powerful than point processes. The principal neighborhood process used for halftoning is the popular error diffusion, first introduced by Floyd and Steinberg in 1975 [13].

In this thesis, a new progressive irregular sampling algorithm has been developed to

implement halftoning using a stochastic screen.

1.5 Outline of Thesis

There are seven chapters in this thesis. This first chapter is an introduction to the topic of image compression using irregular sampling and scattered data interpolation, and halftoning. Chapter 2 gives an overview of existing irregular sampling algorithms in the literature, namely Farthest Point Sampling (FPS) and skewness based sampling. Chapter 3 introduces two novel progressive irregular sampling algorithms - *gaps* and Faster Farthest Point Sampling (FFPS), both based on ideas from FPS. Test results comparing the existing and new methods are also included in this chapter.

Chapter 4 presents a literature review on recent scattered data interpolation techniques, namely adapted 4-Nearest Neighbor Interpolation (adapted 4-NNI) and Multilevel B-Spline Approximation (MBA). A new scattered data interpolation technique based on MBA called New Edge-Directed Multilevel B-Spline Approximation (NEDMBA) is presented in Chapter 5 to improve the appearance of edges in the reconstruction of *gaps*, FFPS and skewness-based samples by MBA. A description of image inpainting using the Mumford-Shah model is included in this chapter as it is utilized by NEDMBA. As well, results of testing of NEDMBA against the previously existing algorithms described in Chapter 4 are given in this chapter.

In Chapter 6, details of available methods for ordered-dither halftoning, the Modified Blue Noise Mask (MBNM), the Void and Cluster (VAC) method (both of which are stochastic screen algorithms) and Linear Pixel Shuffling (LPS) halftoning (a dispersed dot ordered-dither algorithm) are given. Then the application of FFPS to the problem of halftoning is described. A new halftoning algorithm called Farthest Point Halftoning is explained and testing results and comparisons are made with the existing standards. Finally, Chapter 7 presents conclusions based on the new results in Chapters 3, 5, and 6 and suggests areas for future work.

Chapter 2

Irregular Sampling - Background

2.1 Introduction

Before discussing irregular sampling, it is important that it be motivated, and so it should be pointed out why regular sampling can be so poor. First the notion of sampling is reintroduced (actually resampling, since a digital image is already sampled regularly in space with period in both dimensions the distance between two adjacent pixels). Suppose the entire image is I, and that the image intensity at the pixel (x, y) is $I(x, y)$. By sampling, it is meant that instead of representing the entire image I with the intensities of all of I's pixels, only some of the information, $\{(x_i, y_i, I(x_i, y_i)), 1 \leq i \leq N\}$ is preserved, where $N \leq N_I$, the total number of pixels in the image. Sometimes, it may be possible not to have to keep the information about the sample positions, and only record the sample intensities. In general, if

$N < N_I$, then we would expect that there would be some loss of information.

As mentioned in Chapter 1, regular sampling can lead to the appearance of aliasing in the reconstructions. In order to partially eliminate this phenomenon, an image can be sampled irregularly, i.e. not on a regular grid. This can be done uniformly, meaning that no region of the signal is preferred, for example with a jittered grid, where some randomness is added to the sample positions of a regular grid, or with a Poisson disk distribution [14]. This has been shown in practice to reduce aliasing at the cost of increased noise. In fact, it can be shown that aliasing and noise satisfy a Heisenberg relation - below a certain limit, a decrease in one quantity is necessarily translated into an increase in the other [15]. Noise, especially high frequency "blue" noise, is less objectionable than aliasing to the human eye, so irregular sampling is a useful strategy for creating improved image reconstructions.

For any signal, however, more information is carried in regions with high variation, for instance edges of an image. If these areas are sampled more densely, it is possible that less information would be lost. Such sampling is called adaptive sampling, whereas sampling which does not depend on the signal is non-adaptive. Image compression and irregular sampling techniques can also be divided into two other categories mostly related to the transmission of images:

- Progressive compression/sampling : An approximation to the original image can be obtained if any prefix of the complete compressed file is used or the

transmission stopped at any point. The approximation grows more accurate as the compression/transmission process is stopped later, as there is a greater amount of pertinent information. In an irregular sampling context, only the first K samples are taken, where K is smaller than the total number of irregular samples.

- Non-progressive compression: If compression is stopped at any point, the reconstruction from the file or transmitted sequence is not an adequate representation of the entire image. While this is the case with the standard GIF format, increasingly, file formats are becoming progressive because of its advantages.

Progressive image compression/transmission is very desirable; for example on the Internet, the user can stop the loading of an image if he/she recognizes what it is and does not require any further detail.

In an uncompressed image, it is possible for the pixel values to be stored in raster scan order. In general, however, when we take samples from an image, in addition to the intensities of the samples, it is necessary for the positions of the samples to also be transmitted. This decreases the compression ratio and bandwidth requirements for transmission. If an image is regularly subsampled, then the positions of samples do not have to be recorded if the starting position and the sampling periods (in each of the two dimensions) are transmitted. It is possible for much more complex and superior algorithms to exhibit the same property. This leads to two categories of

sampling algorithms: those requiring a point map, and those that do not, with those that do not clearly being favored.

In this chapter, two existing irregular sampling methods for digital images are introduced - Farthest Point Sampling and skewness-based sampling. Farthest Point Sampling is a progressive image sampling strategy that does not require a point map. As will be seen, this strategy does not produce acceptable results for all images or all reconstruction algorithms. Skewness-based sampling is much faster, but unfortunately is not progressive in its published form, needs input parameters specific to the image being sampled, and requires a point map. A progressive version of skewness-based sampling is presented at the end of this chapter.

2.2 Farthest Point Sampling

In one dimension, in sampling a function on a finite length segment, it can be shown that the best progressive sampling strategy after n samples have been selected (where "best" means the one that leads to the least amount of information being lost from the original signal) is to choose the next sample (sample $n+1$) to be in the middle of the largest unsampled portion of the line segment [14]. This is assuming a stationary image model, meaning that the basic characteristics of the image do not change with position. This however is unrealistic for many real-world signals, especially images, and this strategy has to be modified as described on page 33. Such a characterization

of the best sampling strategy is not possible to obtain in closed form for two dimensional signals [14], but by extension, the same process can be used in two dimensions and there is some confidence that it will still give good results.

Definition of Voronoi Diagram Given a set of points $P = \{p_i\}_{i=0}^{N}$ in 2-D space, the *Voronoi diagram* of P is a partition of the plane into tiles T_i such that T_i is a polygon, not necessarily bounded, containing all points which are closer to point p_i than to any of the other points p_k with $k \neq i$. A tile T_i is said to be *contiguous* with T_j if both T_i and T_j share a common edge. A *vertex* of a Voronoi diagram VD is any vertex of any tile of VD.

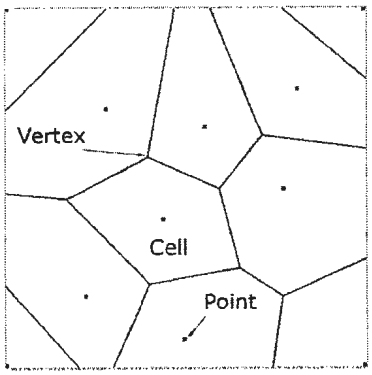


Figure 2.1: A simple labeled Voronoi diagram

In practice, only bounded Voronoi diagrams are taken into consideration. These are formed when some bounded region R is divided into cells (i.e. tiles) and the plane outside of R is not looked at. This is done because the Voronoi diagrams will be taken over an image, which only has a finite extent. An example of a bounded Voronoi diagram with some of the relevant terms labeled is given in Figure 2.1. By

a theorem proven in [14], the point in R farthest from all points in a set of sample points S is guaranteed to be a vertex of the Voronoi diagram of S. This suggests that if $S_N = \{s_i\}_{i=0}^{N}$ is a set of samples, the Voronoi diagram VD_N of S_N may be constructed, and then each of its vertices can be checked in turn to find the point which is the farthest from S_N. This vertex can then be selected as the subsequent sample for a progressive sampling of R. Since some initial configuration has to be started with, samples at the four corners of the image R and at a random point within the image are chosen. After selecting the next sample s_{N+1}, the set of samples is $S_{N+1} = S_N \cup s_{N+1}$, and the new Voronoi diagram VD_{N+1} of S_{N+1} must be created. VD_{N+1} could just be generated from scratch, but in fact the addition of s_{N+1} only changes VD_N in a local manner, meaning that the information in VD_N can be used for the creation of VD_{N+1}. The method to do this was outlined in [16], and is described in more detail here. The theoretical discussion which follows is made more concrete by an example starting from page 29.

Description of Incremental Voronoi Diagram Construction

Definition of contiguity list For each point p_i in the Voronoi diagram, we maintain a *contiguity list*, $CL(p_i)$, which holds the information of the other points in the diagram to which p_i is adjacent. A point is considered to be adjacent to another point if their respective tiles share an edge. The points in this list are kept in counter-clockwise order around p_i, making it easier to update the contiguity list when adding

a new vertex. The Voronoi diagram of a set of points can be fully recovered from the set of contiguity lists of its points, because each edge of a tile is accounted for by the contiguity between the two points on either side of the edge. In fact, this edge is the perpendicular bisector of the line joining the two points.

Elaboration of constraints Since only bounded Voronoi diagrams are being dealt with - in this case Voronoi diagrams bounded by the extent R of the image we are sampling - constraints, corresponding to the four lines forming the boundary of the rectangular region R, must also be taken into account. These constraints may appear in the contiguity lists of points, because the boundary of a vertex's tile can include part of a constraint if the tile is on the edge of R. Moreover, the four line constraints $\{c_i\}_{i=-4}^{-1}$ are permitted to have their own contiguity lists, so they are treated as points in their own right. The constraints are indexed with negative integers so that all objects in the VD are uniquely identified by their index, where objects are either samples or constraints. This generalization of allowing the constraints to have contiguity lists of their own makes updating the Voronoi diagram upon the addition of new vertices more straightforward. Whenever a new sample s is added, and the tile which is built up for this new sample has an edge on a constraint, the constraint can be thought of as being a *virtual* point, the point being located at the reflection of s about the constraint. Observe that the position of this virtual point depends on the location of the sample s we are adding. In this way, the concept of contiguous points

can be generalized by the following. For all adjacent objects to s, the boundary between s and the neighboring object is made up of the perpendicular bisector of s and either:

1. the object itself if it is a point or

2. the virtual point associated with the object if it is a constraint.

Updating VD_N to VD_{N+1} The procedure for updating a Voronoi diagram VD_N to create VD_{N+1} upon the addition of s_{N+1} follows. A thorough description is included here since this full description of the updating of contiguity lists is not given in reference [16] where this incremental algorithm was first presented, and doesn't appear to be included anywhere else in the literature. The contiguity list of s_{N+1}, $CL(s_{N+1})$, is built up in clockwise order, and s_{N+1} should be inserted into the contiguity lists of objects with which it ends up being contiguous. First, find the point σ_c of VD_N which is the closest to s_{N+1}. The notation σ_c is used because it starts with the same letter ('s') as sample, but it may also be a virtual point if the closest object to s_{N+1} is a constraint. Let $\sigma_0 = \sigma_c$. σ_0 is guaranteed to be in $CL(s_{N+1})$, so that is why we start by considering it. Initialize the contiguity list of s_{N+1} to be empty. Go through the next 5 steps for each σ_i, where we start with $i = 0$, and increment i with each loop:

1. Take the perpendicular bisector PB_i of s_{N+1} and σ_i (where σ_i can be a virtual point).

2. Go through all the objects in the contiguity list of σ_i, $CL(\sigma_i)$ and choose the one σ_{i+1} whose perpendicular bisector with s_{N+1} crosses PB_i the earliest, keeping in mind that we are traversing the boundary of the tile of s_{N+1} in a clockwise direction. The determination of "earliest" is given below. As with most computational geometry algorithms, we must make comparisons between coordinates of points, and since these can be very close to each other, for the incremental Voronoi diagram construction, we used the `long double` type. In the implementation we used, the `long double` type took up 12 bytes, and so is more precise than the standard `double` type when distinguishing between near but distinct values. A `long double` value v_1 was considered to be greater than another v_2 if $v_1 > v_2 + \epsilon$, where we chose $\epsilon = 2 \times 10^{-11}$. This value of ϵ was selected because if it is too small, rounding errors may cause two equal points to be regarded as distinct, while if it is too large, then two distinct points may be considered to be equal. The latter case may cause problems because when going around the new tile, we look for the next distinct vertex of the new tile. If the next vertex is very close to the previous one and is regarded as equal, then perhaps there will be no next distinct vertex, and the construction of the new tile cannot continue. Now we move on to the description of finding σ_{i+1}, the next object in our march around the new tile.

   - If σ_i is strictly to the right of s_{N+1} (using the above ϵ value), then we

let σ_{i+1} to be the element of $CL(\sigma_i)$ whose perpendicular bisector with s_{N+1} intersects PB_i below the midpoint of a line between s_{N+1} and that element of $CL(\sigma_i)$, and the closest to this line's midpoint. To be strictly to the right of s_{N+1}, the x-coordinate of σ_i must be greater than that of s_{N+1}.

- If σ_i is at the same horizontal position as s_{N+1}, then there are two cases depending on whether σ_i is below s_{N+1} or above it (using the ϵ value). If it is below, then we choose as σ_{i+1} the element of $CL(\sigma_i)$ whose perpendicular bisector intersects PB_i the closest to and to the left of the midpoint of a line between s_{N+1} and that element. If it is below, we choose the one whose perpendicular bisector intersects PB_i the closest and to the right of this midpoint.
- Finally, if σ_i is strictly to the left of s_{N+1} (once again using ϵ), then we let σ_{i+1} be the element of $CL(\sigma_i)$ whose perpendicular bisector with s_{N+1} crosses PB_i above and closest to the midpoint of s_{N+1} and that element of $CL(\sigma_i)$.

3. Now since s_{N+1} is adjacent to σ_i, it must be inserted into its contiguity list.
   - If σ_i is a virtual point while σ_{i+1} is a regular point, insert s_{N+1} in the contiguity list of the constraint associated with σ_i *after* σ_{i+1}.
   - Otherwise, insert s_{N+1} in the contiguity list of σ_i *before* σ_{i+1}.

4. Insert σ_i at the end of the contiguity list of s_{N+1}.

5. If $i > 0$, σ_{i-1} and σ_{i+1} (or the defining constraint if one is a virtual point) will sometimes no longer be contiguous because s_{N+1} is now between them. If σ_{i-1} and σ_{i+1} were indeed contiguous, then wait until the entire tile of s_{N+1} has been formed. The contiguity is deleted if and only if the edge forming the border between σ_{i-1} and σ_{i+1} lies entirely in the new tile of s_{N+1}. If this ends up being the case, then σ_{i+1} is deleted from the contiguity list of σ_{i-1} and σ_{i-1} from the contiguity list of σ_{i+1}. Before performing this deletion, also check that the boundary of the Voronoi cell of s_{N+1} is not complete and a triangle, because if this is the case, the adjacency of σ_{i-1} and σ_{i+1} should be maintained.

6. Repeat steps 1 to 5 until $\sigma_{i+1} = \sigma_0$.

It is necessary to verify that the old boundary edge between the objects whose contiguity is being deleted (o_1 and o_2) lies completely within the tile formed by the newly added sample, before the contiguity is deleted. If it does lie completely within the new tile (and the two were indeed contiguous before), then more contiguities have to be considered for deletion.

Call the following rule (consisting of this entire paragraph) D: If the contiguity between o_1 and o_2 was deleted, then o_1 had to be in $CL(o_2)$ and o_2 had to be in $CL(o_1)$. If the element before o_1 (remembering the lists are circular) in $CL(o_2)$ is not s_{N+1}, then consider the contiguity between o_2 and this object for deletion, otherwise

consider the contiguity between o_2 and the element two before o_1 in $CL(o_2)$. The same is done for the element one after or two after o_1 in $CL(o_2)$, and symmetrically consider the contiguities between o_1 and the elements one or two before and after o_2 in $CL(o_1)$.

Then the contiguity list for the new sample is in clockwise order, so it is reversed to make it conform to the order of the previously existing contiguity lists of VD_N. The contiguity between the first object in the contiguity list of s_{N+1} and σ_1 (now the second last object, after reversal) is deleted in s_{N+1}'s contiguity list, because of the cyclic nature of the list, and since the information was initially unavailable about the last element of the list when we were looking at σ_0. Once again, the conditions for non-deletion are checked.

Example of Incremental Voronoi Diagram Construction

This algorithm is clarified through the use of an example of the addition of a seventh point to an already constructed bounded Voronoi diagram VD_5 of six points. Before the addition of the 7th point (numbered sample 6 in Figure 2.2 since sample 0 is the first sample), the contiguity lists of all the samples and constraints are as given in Table 2.1. In that table, the constraints are labelled with negative integers, as in Figure 2.2. Samples are labelled with non-negative integers, and only the indices are included in the lists. Also note that the first four samples are located at the corners of the image region R, as per the convention.

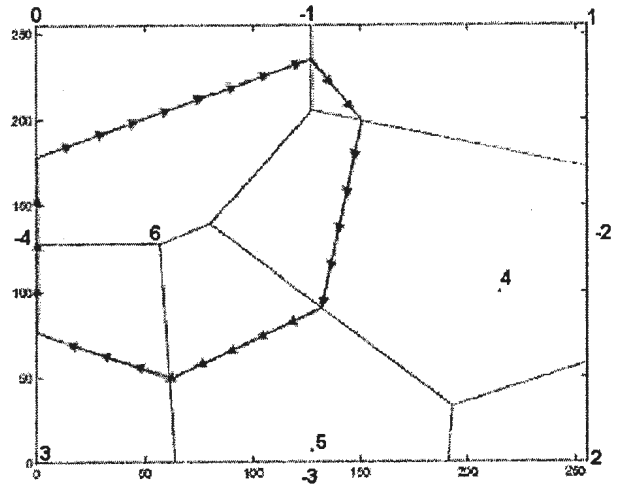


Figure 2.2: Incremental Construction of Voronoi Diagram using Green and Sibson's Method

Constraint/Sample Index	Contiguity List
-4	-1,0,3,-3
-3	-4,3,5,2,-2
-2	-3,2,4,1,-1
-1	-2,1,0,-4
0	-4,3,5,4,1,-1
1	-1,0,4,-2
2	-2,4,5,-3
3	-3,5,0,-4
4	1,0,5,2,-2
5	2,4,0,3,-3

Table 2.1: Contiguity lists for objects in Figure 2.2 before Sample 6 added

The new sample (number 6) is closest to constraint -4, so σ_0 is the reflection of s_6 in the left-handed constraint. PB_0 is the part of constraint -4 marked with arrows in Figure 2.2. The object in the contiguity list of constraint -4 whose perpendicular bisector with s_6 crosses PB_0 the first, going upwards, is s_0, which becomes σ_1. Because σ_0 was a virtual point, insert s_6 into the contiguity list of constraint -4 *after*

Constraint/Sample Index	Contiguity List
-4	-1,0,6,3,-3
-3	-4,3,5,2,-2
-2	-3,2,4,1,-1
-1	-2,1,0,-4
0	-4,6,1,-1
1	-1,0,6,4,-2
2	-2,4,5,-3
3	-3,5,6,-4
4	1,6,5,2,-2
5	2,4,6,3,-3
6	3,5,4,1,0,-4

Table 2.2: Contiguity lists for objects in Figure 2.2 after Sample 6 added

0. PB_1 is the perpendicular bisector of s_6 and s_0, and the object in $CL(\sigma_0)$ whose perpendicular bisector crosses PB_1 first is s_1. $\sigma_1 = s_0$ is a sample, so s_6 is inserted in $CL(s_0)$ (contiguity list 0) before s_1 (1). This is σ_2, so the contiguity between σ_2 and p_0 has to be considered for deletion. This is done after the formation of the new tile.

The rest of the formation of the tile of s_6 is not too complicated. From here on, only the indices of the points and constraints are used, and $CL(i) = CL(s_i)$ for $i \geq 0$, and $CL(i) = CL(c_i)$ for $i < 0$. Four is the object in $CL(1)$ whose perpendicular bisector with 6 crosses PB_2 first. So insert 6 in contiguity list 1 before 4. The contiguity between 0 and 4 has to be considered for deletion. Then, insert 6 in $CL(4)$ before 5, and consider the contiguity between 5 and 1 for deletion. Continuing on, insert 6 in $CL(5)$ before 3, and mark the contiguity between 3 and 4 as a candidate for deletion. The next object in the march around the new tile of

s_6 is -4, and since this was the closest object to s_6, the formation of the new tile is complete. Insert 6 in $CL(3)$ before -4 and consider the contiguity between 5 and -4 for deletion. Every neighboring object that is encountered is inserted into $CL(6)$ in order, so that we get $CL(6) = -4, 0, 1, 4, 5, 3$, after which it is reversed. Also the contiguity between 3 and 0 has to be considered for deletion.

Now it only remains to check which contiguities among the contenders have to be deleted. The contiguity between σ_2 and σ_0, namely -4 and 1 does not have to be deleted, since they were not contiguous to begin with. The contiguity between 0 and 4 is deleted because the edge between the tiles of the two samples lies entirely within 6's new tile (see Figure 2.2). This means 4 is removed from $CL(0)$ and 0 is removed from $CL(4)$. By Rule D, contiguities between 0 and 5, 0 and 1, 4 and 5, and 4 and 1 have to be considered. These additional possibilities are added to the set of contiguities that are considered, and the original candidates are now looked at more closely. 5 and 1 were not contiguous, so this adjacency doesn't have to be removed. The same is the case with 3 and 4, and 5 and -4. Finally, 3 and 0 were contiguous and the edge between the two lies entirely within the new tile of s_6, so this contiguity is eliminated. By rule D, contiguities between the pairs of objects 3 and 5, 3 and -4, 0 and -4, and 0 and 5 have to be looked at. None of these contiguities are categorized as having edges completely within the new tile, except that between 0 and 5, which was one of the pairs to be considered by the first use of rule D. So it is only necessary to examine the pairs of objects which were candidates from the initial application of

rule D. Of these pairs, 0 and 5 is the only one which has to be deleted. Another (recursive) use of rule D is needed, but this does not add any more deletions. After performing all of the above steps, the contiguity lists of all the objects are as shown in Table 2.2, and the same procedure is followed for the addition of the next sample, sample 7, which is necessarily a vertex of this new updated Voronoi diagram VD_6.

Adaptive Farthest Point Sampling As was mentioned at the beginning of this section, the Farthest Point strategy is only valid for a stationary model of the image being sampled, and is not good for typical images, where we would want more samples closer to details and fewer in smooth regions. Farthest Point Sampling on its own does not change the sample density depending on the image but does clearly break up the regular grid of regular sampling. To take into account the characteristics of the image being sampled, a weighted distance was proposed in [14]. This distance is defined to be as follows. If v is any vertex of the Voronoi diagram at any stage, then its weighted distance from all samples can be calculated by

$$W(v) = R^2(v) \max_{s_i, s_j \in N(v)} (B_{min}(s_i, s_j)).$$

$N(v)$ is the set of the three nearest samples to v taken so far, and $R(v)$ is the minimum distance from v to the samples taken so far. B_{min} is a lower bound on the

bandwidth, calculated by

$$B_{min}(p,q) = \frac{\arcsin(\frac{2I(q)-M}{M}) - \arcsin(\frac{2I(p)-M}{M})}{2\pi d(p,q)}.$$

$I(p)$ is the image intensity at point p in the image and M is the maximum intensity permitted in the image, for example 255 for 8-bit grayscale. $d(p,q)$ is the Euclidean distance between points p and q and it is assumed without loss of generality that $I(q) \geq I(p)$. $N(v)$ can be easily found. Suppose the weight of a vertex v is being calculated. If it is defined by three samples (and is thus their circumcentre), then these three must be the elements of $N(v)$. Otherwise, v must lie on a constraint. It has been proven by this author that $N(v)$ is contained in a very limited subset of the samples in the entire Voronoi diagram. The point v is a vertex, and so is a corner of a set of tiles T_v. Then only the set of samples whose tiles border T_v, along with the samples whose tiles are in T_v have to be checked. The three closest samples must be included in this set of samples.

Now the motivation for this formula for the weight of a vertex is examined more closely. Obviously, the further a vertex is away from the previous samples, the higher the weighted distance will be, due to the presence of the $R^2(v)$ factor. The higher that $I(q)$ is, the greater the first arcsin term will be in $B_{min}(p,q)$, and the lower that $I(p)$ is, the lower the second arcsin term in B_{min} will be. So the greater variation there is in the intensities of pixels in $N(v)$, the greater $B_{min}(\cdot,\cdot)$ will be, and thus the

greater that $W(v)$ will be. So regions with high variation in the image will be sampled more densely than uniform regions. At each step of the incremental construction of the Voronoi diagram, the vertex with the maximum weighted distance is chosen. To do this, it is necessary to go through all of the vertices of the Voronoi diagram. Only contiguities are recorded - to get the vertices, the generalized circumcentres are computed of any object and any two consecutive items in its contiguity list, where the first and last elements of a contiguity list are also considered to be consecutive. By generalized circumcentre, it is meant that virtual points are substituted for constraints. Given three points, $\{\mathbf{p}_i = (x_i, y_i),\ 1 \leq i \leq 3\}$, the circumcentre (x_C, y_C) is given by [17]:

$$y_C = \frac{\{(x_3^2 - x_1^2) + (y_3^2 - y_1^2)\}(x_2 - x_1) + \{(y_2^2 - y_1^2) + (x_2^2 - x_1^2)\}(x_1 - x_3)}{2(y_3 - y_1)(x_2 - x_1) + 2(x_3 - x_1)(y_1 - y_2)}$$

$$x_C = \frac{(y_2^2 - y_1^2) + (x_2^2 - x_1^2) + 2(y_1 - y_2)y_C}{2(x_2 - x_1)}$$

When the weighted distance is used to select the next vertex in FPS, the sampling method is known as adaptive Farthest Point Sampling as it depends on the image. Simply selecting the farthest point at each step without reference to any image is referred to as nonadaptive or uniform Farthest Point Sampling. From now on, if it is not explicitly stated, the adaptive algorithm is assumed. Finding the vertex of the Voronoi diagram with the largest weight is potentially a very slow operation if the naive approach is taken and all of the vertices of the Voronoi diagram are searched

through at each iteration. Instead, a great improvement in speed can be made if a balanced binary tree of all the vertices in the Voronoi diagram is used; this can be accomplished by using the `set` container in STL [18] (The C++ Standard Template Library written by Silicon Graphics), sorted in decreasing order by vertex weight, since `set` uses a red-black balanced binary tree to store its elements.

Although the reconstructions from samples obtained from Farthest Point Sampling are not necessarily as good as those from other sampling methods, one advantage of FPS over other sampling techniques - for example skewness-based sampling which is described next - is that it does not require the transmission of the point set. As long as the position of the first non-corner random sample is known, the rest of the samples are uniquely determined. This is obvious for the nonadaptive version of FPS, but it also holds for the adaptive version. This is because the points in the constructed Voronoi diagram VD_N at any stage N are only previously selected samples. Thus, the weighted distances of each of the vertices of the diagram, depend only on the distance of the vertices to the previous samples, and the intensities and distances between the three closest samples to each of the vertices. After transmitting the gray-levels of the four corners and the position and gray-level of the first random point, it is only necessary to transmit the intensities of the subsequent samples.

An example of taking 3035 samples of the `zelda` image with adaptive Farthest Point Sampling is given in Figure 2.3, where the samples are overlaid on the original image as white dots. These samples represent about a 21.6:1 compression ratio.

Figure 2.3: 3035 Farthest Point samples of `zelda`

2.3 Skewness-based Sampling

2.3.1 Basic algorithm

In 2001, in response to the supposed high complexity of Farthest Point Sampling, Ramponi and Carrato devised a new irregular sampling algorithm based on local skewness [11]. This sampling algorithm, however, was designed primarily for reconstruction based on 4-Nearest Neighbor Interpolation (4-NNI) which is described in the same paper. As well, unlike Farthest Point Sampling, it is not truly progressive (samples are not generated one at a time), so it is not possible to control *a priori* how many samples will be taken. Finally, certain parameters for the algorithm are image specific and must be chosen at least partly based on trial and error and on previous knowledge of the image characteristics.

In the field of statistics, the sample skewness of a set $X = \{x_i, 1 \leq i \leq N\}$ is defined to be $sk(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^3$, where μ is the mean of the elements of X. For an image I, the local sample skewness is based on a 2-D mask centered on a point. The mask is a set of points centered on the mask origin (0,0) which specifies which points around (i, j) are used to compute the skewness. For this application, we use a 5x5 square mask, so that $M = \{(i, j)|i, j \in \mathbb{Z}, |i| \leq 2 \text{ and } |j| \leq 2\}$, and the sample skewness of the point (i, j) is $sk((i, j)) = \frac{1}{|M|}\sum_{(i',j')\in M}(I(i + i', j + j') - \mu(i, j))^3$.

It turns out that pixels in an image which are either close to or exactly on edges have skewnesses with high absolute values. For 4-NNI, in order to retain the most information about the original image with the fewest number of samples, it is not desirable to have any samples in uniform or linearly varying areas, or exactly on edges, but it is desirable to have many samples close to edges and in heavily textured regions. These specifications correspond exactly to pixels with high absolute skewness values except for those precisely on edges, which may also have high skewness magnitudes.

Since the goal is to obtain sampling grids with image-dependent concentrations, pixels with low skewness absolute values are sampled more sparsely than those with high skewness absolute values. To expand upon this, a multiresolution approach is used, with differing skewness thresholds and radii of exclusion for each level. A sampling grid from level 1 to level n is built up. First, the skewness $sk(i, j)$ of each pixel (i, j) is calculated and then normalized to $\bar{sk}(i, j) = \frac{sk(i,j)}{\max_{(i',j')\in I} sk(i',j')}$, where I is the entire image. At level l, all pixels which have $\theta_{l-1} < \bar{sk}(i, j) < \theta_l$

$(1 \leq l \leq n)$, where the θ_k's are skewness thresholds, are taken, but only if they satisfy another condition. They must satisfy a radius of exclusion r_l, specific to the level l. This means that the sample that is chosen next is at least a distance of r_l away from all samples already selected, either in lower levels or in the current level. Because samples in the same level are considered when accounting for the radius of exclusion, some ordering for the current level's samples must be taken - a raster scan order is chosen for its simplicity, and Ramponi and Carrato report that this doesn't have much of a detrimental effect on the results, despite the fact that it is not as general as possible. An additional limitation on the samples is that they must have gradient magnitude (measured for example by the Sobel operator, see Section 3.1.2 for more details) less than a threshold θ^g, given below The skewness thresholds θ_l's are an increasing sequence of numbers with $\theta_0 = 0$ and $\theta_n = 1$. The r_l's are strictly decreasing positive numbers.

At each level l, a raster scan is performed and a sample chosen only if it satisfies the gradient, skewness and radius of exclusion requirements. Because the r_l's are decreasing and the θ_l's increasing, pixels with higher skewnesses (except those precisely on edges) are sampled more densely than those with low skewness magnitudes, as desired. The net effect of this sampling algorithm is to sample linearly varying and uniform regions very sparsely, with double rows of samples around the contours of the image. An example is given in Figure 2.4, where the double rows can be clearly seen. Note that the same number of samples are taken as in Figure 2.3. The

parameters for this sampling are $n = 3$, $\theta_s = [0.0015, 0.004, 0.008]$, $r = [15, 11, 7, 2]$, and $\theta^g = 128$.

Figure 2.4: 3035 Skewness-based samples of `zelda`

2.3.2 Progressive Skewness-based Sampling

As stated previously, one of the drawbacks of the skewness-based sampling given in [11] is the fact that despite its multiresolution nature, samples are not chosen progressively. All the samples at a given level have to be transmitted before the next higher resolution image can be reconstructed because the samples in any given level are generated in raster scan order. Using Linear Pixel Shuffling [19], it is possible to remedy this problem. It is proposed in [11] that the bitstream be selected in a pseudo-random sequence, but this is not followed up on; using LPS as a method by which the weaknesses of raster scan sampling can be overcome is proposed for the

first time here. The aim of the LPS method by Anderson is to efficiently order the pixels in an image so that the image is "jumped" about; pixels which are close to each other are far apart in the ordering. The details are now given.

Let $G_0 = 0$, $G_1 = G_2 = 1$ and $G_k = G_{k-1} + G_{k-3}$ for $k > 2$. The negative indices of G can be determined by the same rearranged Fibonacci-like recurrence equation, namely $G_{k-3} = G_k - G_{k-1}$. If the input image is r rows by c columns, we define a square table with sides G_n ($G_n \geq \max(r, c)$) such that the entire image can be fit inside the table. T_{pq} is set to $(pG_{n-2} + qG_{n-1}) (\mathrm{mod}\ G_n)$. It can be shown that values that are close in number are far apart in distance in the table. So if the pixels with entry 0 in the table are sampled first, then those with entry 1, etc., all parts of the image will be visited without discriminating against any particular section. To go through the pixels in this order, the matrix $M = \begin{pmatrix} G_{-n+1} & G_{n-3} \\ G_{-n} & G_{n-2} \end{pmatrix}$ is used. Simply perform the following algorithm to change the raster-scan ordering of the skewness-based samples to a progressive scheme (in the code ' indicates transposition):

```
for i = 0 to G_n do
   for j = 0 to G_n do
      (x, y)' = M*(i, j)';
      if (x, y)' is a sample from skewness-based sampling,
         take (x, y)' as the next LPS sample.
      end if
   end do
end do
```

2.4 Conclusions

In this chapter, two previously existing irregular sampling algorithms - Farthest Point Sampling and skewness-based sampling were presented. Farthest Point Sampling has two forms - non-adaptive and adaptive. The sampling densities of both adaptive FPS and skewness-based sampling conform to local image characteristics; basically portions of the image with high variation are sampled more densely than those which are more uniform. Irregular sampling grids resulting from these methods for some test images will be given in Chapter 3, compared with the results of new progressive irregular sampling methods developed in that chapter. The details of an improvement to skewness-based sampling, making it progressive, were also given in this chapter.

Chapter 3

New Progressive Irregular Sampling Methods

3.1 The *gaps* method for irregularly sampling an image

3.1.1 Introduction

This section presents a new progressive irregular sampling technique called *gaps* which was developed to give good reconstructions, and to be reasonably computationally efficient. The originating motivation was to approximately emulate unoptimized Farthest Point Sampling by means of a less computationally complex method. The idea behind the *gaps* method is that at any stage in the progressive sampling, samples already chosen influence the closest distance of a pixel to any sample only

for pixels close to them. In other words, if a sample is far away from a pixel, then it is unlikely that it is the closest one to that pixel and so can be safely ignored. The *gaps* algorithm can be run, like FPS, in both adaptive and non-adaptive modes. The following section contains a detailed description of *gaps* with experimental results given in Subsection 3.4.2.

3.1.2 Algorithm Description

Calculation of gap magnitudes

As noted in the introduction, the simplifying assumption of *gaps* is that a given sample is the closest sample to a position in an image only for positions that are close by. For simplicity, we can say that a sample is close to a pixel if it is within a certain distance horizontally or vertically from it. So around each already chosen sample, a cross structure is created, as shown in Figure 3.1.

Each of the four lines in the cross is of length $\max_{(i,j)\in I} G(i,j)/\sqrt{2}$, where $G(i,j)$ is defined below to be the gap magnitude of pixel (i,j); the gap magnitude function $G(i,j)$ varies with iteration number. I is the set of pixels in the entire image. G varies with iteration number because as samples are chosen, the maximum gap magnitude changes if the new sample cuts off the previously existing gap of maximum magnitude.

To implement the method, linked lists of column numbers of samples influencing

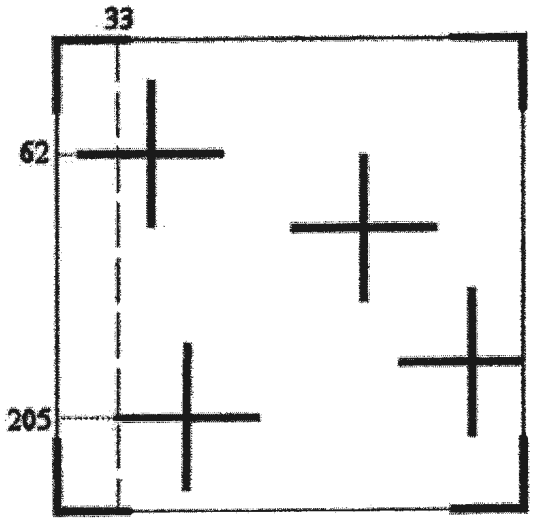


Figure 3.1: Intermediate state of *gaps* algorithm for a 256x256 sized image

each row are stored for each row of the image while linked lists of row numbers of samples influencing each column are stored for each column of the image. R_i is the row list for the ith row, while C_j is the column list for the jth column of the image. For example, in Figure 3.1, the list for column 33, C_{33}, has entries containing row numbers 0, 62, 205 and 255.

In a manner similar to the Farthest Point Sampling (FPS) algorithm, the four corner pixels are selected as the first four samples. The horizontal and vertical gap values, which roughly measure how far away each pixel is to the closest sample horizontally or vertically, are then calculated for each pixel in the image. To find the horizontal gap value of the pixel (i, j) the row list R_i is examined to find which two samples S_1 and S_2 in the row list R_i, that pixel (i, j) lies between. Then the perpendicular bisector of these two samples is taken. Which of the two samples S_1 or

S_2 lies on the same side of the perpendicular bisector as the pixel (i,j) is determined and the horizontal gap value $G_H(i,j)$ is calculated to be the Euclidean distance of this closest sample to the pixel (i,j). A similar procedure is used to find the vertical gap value $G_V(i,j)$ for each image pixel. The overall gap magnitude $G(i,j)$ of the pixel (i,j) is defined to be $G(i,j) = \min(G_H(i,j), G_V(i,j))$. In fact, the calculation of G_H can be more easily made by letting $G_H(i,j) = \min\{d((i,j),S_1), d((i,j),S_2)\}$ since this encapsulates the bisection and nearest point operations described above. Here, as before, $d(\mathbf{x},\mathbf{y})$ denotes the Euclidean distance between the pixels $\mathbf{x}$ and $\mathbf{y}$. In Figure 3.2, $G_H(i,j) = \min(G_{H1}, G_{H2})$ and $G_V(i,j) = \min(G_{V1}, G_{V2})$, so that $G(i,j) = \min(G_{H1}, G_{H2}, G_{V1}, G_{V2})$.

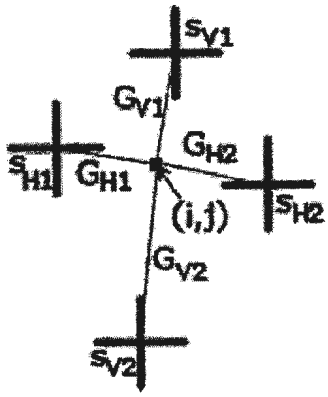


Figure 3.2: Calculation of the gap magnitude of a pixel (i,j)

A special case occurs when R_i or C_j is empty, or when the first entry of one of these lists is not 0, or the last entry is not equal to the maximum possible value (the width or height of the image for row and column lists, respectively). If a row or

column list falls into one of the last two cases, it is said *not to have a proper endpoint.* For simplicity, suppose that a row list is being considered. Define the row group γ_i for an empty row i to be the set of adjacent consecutive rows with lists which are empty, and γ_i for a row with no proper endpoint to be the set of adjacent consecutive rows with lists which have the same first or last entry as row i. Those adjacent rows with the same first entry are taken if the first entry of the row under consideration is not 0 or those with the same last entry if the last entry of the row being considered is not equal to the maximum possible value, namely the width. The intersection of both these sets is taken if the row list has neither first entry 0 or last entry the width of the image.

A new parameter for the algorithm, $fillfactor$, must be introduced. The closest sample which is within $fillfactor \times \max_{(i,j)\in I} G(i,j)/\sqrt{2}$ of row i is used to fill in γ_i. Suppose that the closest sample to the row is below it. By filling in, we mean that an entry for the closest sample is inserted into each of the row lists which are either empty or up to the top row in γ_i. If any rows continue to have no proper endpoint, these are recursively filled in if possible using the same procedure.

After the filling-in stage, there may still be rows which are empty or which have no proper endpoint. If there are empty rows, the gap magnitude of the pixels in these rows are computed as being the distances to the closest samples. If there are rows with no proper endpoint, then the gap magnitude is taken to be the distance to the sample closest to that row's gap. This sample must satisfy the further condition

that it (the closest sample) is to the left of the first sample in the lists of the rows of γ_i if the row has no proper endpoint on the left, and to the right of the last sample in the lists of the rows of γ_i if the row has no proper endpoint on the right. After this procedure, all rows and columns have correctly defined row and column lists respectively.

Next, in the non-adaptive scheme, the weight W of each pixel is defined to be

$$W(i,j) = F\left(\frac{G(i,\,j)}{\max_{(i',j')\in I} G(i',\,j')}\right) \tag{3.1}$$

where typical examples of the function $F(x)$ are $\sqrt[3]{x}$ and $\frac{\arctan(11x-3)+\arctan(3)}{2.5}$. The second example is chosen principally because of its inflection point, and we will justify the use of a similar type of function in the paragraph before Equation 3.5. We included a multiplicative factor in [20], i.e. $W(i,j)$ was described by the equation: $W(i,j) = M_G \times F(\frac{G(i,j)}{\max_{(i',j')\in I} G(i',j')})$ but further investigation showed that this is unnecessary as only the relative weights of the image pixels are important.

Adaptive *Gaps*

Unlike non-adaptive *gaps*, in adaptive *gaps*, the edge magnitude of the pixel being considered as well as those in its neighborhood have an effect on $W(i,j)$. This

function can be computed using the general formula:

$$W(i,j) = H(i,\, j,\, l,\, \{G(i',\, j')\}_{(i',j')\in I},\, \{h_{Mask}(i',\, j')\}_{(i',j')\in I}, I(i,\, j)) \qquad (3.2)$$

Notice that $W(i,j)$ depends as well on the iteration number l in the case of this general formula. For the calculation of $W(i,j)$, H should be increasing with respect to $G(i,j)$, since $G(i,j)$ roughly approximates the distance of (i,j) to the closest sample, and the larger this number is, the more likely that this pixel should be sampled next. $h_{Mask}(i,j)$ is a function of the Sobel edge magnitudes in a mask centered on pixel (i,j), and once again H should increase with increasing h_{Mask} for any given position.

We briefly digress to discuss the Sobel operator. Horizontal (S_x) and vertical (S_y) Sobel masks of dimensions 3x3 are shown in Figure 3.3. Each of these masks is convolved with the image. An E_x value is obtained from convolution of S_x with the image and an E_y value is obtained from convolution of S_y with the image. The final gradient magnitude is found by the formula $Sob(i,j) = \sqrt{E_x^2 + E_y^2}$. The higher the calculated gradient magnitude of a pixel is, the stronger is the "edgeness" characteristic of that pixel.

We now return to our discussion of the pixel weights for adaptive *gaps*. One example of a function $h_{Mask}(i,j)$ is $Sob^{Max}_{Mask}(i,j)$, where $Sob^{Max}_{Mask}(i,j)$ is the maximum Sobel edge magnitude in the mask centered on pixel (i,j), so that pixels within the

$S_x =$

-1	0	1
-2	0	2
-1	0	1

(a) Horizontal 3x3 Sobel Mask

$S_y =$

1	2	1
0	0	0
-1	-2	-1

(b) Vertical 3x3 Sobel Mask

Figure 3.3: 3x3 Sobel Masks

mask radius of strong edges are heavily weighted. Another example is $Sob^{Max}_{Mask}(i,j) - Sob(i,j)$. This function tends to weight more heavily pixels which are in smooth regions close to strong edges. This second example for $h_{Mask}(i,j)$ is a good function if the *gaps* samples are to be used with adapted 4-NN interpolation (described in more detail in Section 4.2) because one of the criteria listed in Ramponi and Carrato's paper [11] for sample selection for this type of interpolation is that there should be many samples close to edges but not exactly on them.

Observe that the above weight function depends on l, the number of the sample we are currently choosing. The purpose of this iteration dependence is to increase the adaptivity of the sampling process as it progresses. At the beginning, not much information about any part of the image is known, so sampling should be done rather

regularly, ensuring that no portion of the image is omitted. Only after this do we wish to obtain the edge information. The values i and j come into play because $G(i,j)$ and $h_{Mask}(i,j)$ are extremely important parts of the weight calculation. We call the initial phase where the image is sampled rather regularly the *quasi-adaptive stage*, while the latter phase, the *adaptive stage*. The transition point between the two phases depends on an image-dependent parameter (i.e., an iteration number) called *cutoff* which is described more fully later by Equation 3.11. The weight of each pixel in the adaptive stage has two terms, while the weight of each pixel in the quasi-adaptive stage has three terms, both based on the general form of $W(i,j)$ given by Equation 3.2. We now give and describe the weight expressions for both the adaptive and quasi-adaptive stages, starting with the adaptive stage as its formula is simpler. The exact form of H will be expanded upon for both these stages. The presence of $I(i,j)$ or the image intensity at pixel (i,j) has not been explained - it is present only in the expression of H for the quasi-adaptive stage, and it will become clear why it is needed when we go into further detail of that stage.

Adaptive stage Recall from Equation 3.2 that the weight W of a pixel (i,j) in the image I being sampled depends in general on the gap magnitudes G and the h_{Mask} of all the pixels in I as well as the position of the pixel and the iteration number l. After a certain number of iterations (i.e. after iteration *cutoff*), the form of the

weighting function W for pixel (i, j) can be simplified as follows:

$$W(i,j) = M_G \times F\left(\frac{G(i,\, j)}{\max_{(i',\, j') \in I} G(i',\, j')}\right) + M_E \times E\left(\frac{h_{Mask}(i,\, j)}{\max_{(i',\, j') \in I} h_{Mask}(i',\, j')},\, ERank(i,\, j)\right), \tag{3.3}$$

where M_G is the gap coefficient and M_E is the edge coefficient, with both parameters being experimentally determined, and E and F are functions that we now give. Note the lack of dependence on the iteration (sample) number l for this adaptive stage.

For efficiency reasons, we keep the $E(\cdot)$ term in Equation 3.3, for both the adaptive and quasi-adaptive stages. The $ERank$ function will be elaborated upon from the bottom of page 59, but basically measures how important $h_{Mask}(i, j)$ is relative to the other h_{Mask} values in the image (its rank amongst the sorted h_{Mask} values divided by the total number of pixels in the image). The $E(\cdot)$ term should be larger for large h_{Mask} and $ERank$ values and smaller for small such values. It is also desired that the $ERank$ value has more influence than the raw h_{Mask} value of a pixel, since if a pixel has low $ERank$ and high h_{Mask} it should be weighted less than one with low h_{Mask} and high $ERank$, which is more important overall to the image. To achieve these goals, we let

$$E(x, y) = \sqrt[3]{x + y\sqrt{y}} \tag{3.4}$$

Next, the function F taking the gap magnitude $G(i, j)$ as an argument should not distinguish too much between large and medium sized gaps, and pixels with

extremely small gaps should not be weighted heavily. So we let

$$F(x) = \frac{\arctan(20x - 3) + \arctan(3)}{\arctan(17) + \arctan(3)} \tag{3.5}$$

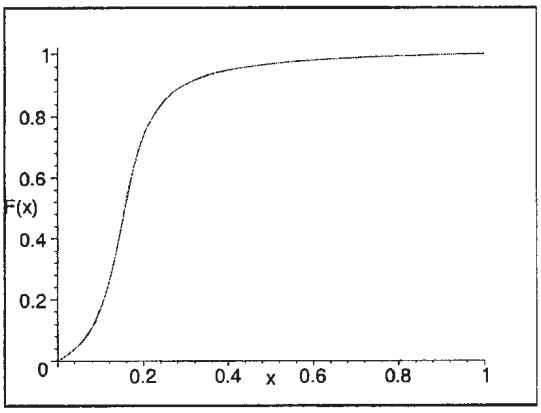


Figure 3.4: Function $F(x)$ used for gap magnitude contribution to pixel weight after iteration *cutoff*

As shown in Figure 3.4, this function $F(x)$ ranges from 0 to 1 on [0,1], while $E(x, y)$ ranges from 0 to 2 on its domain, [0,1]x[0,1]. An alternate formula for $W(i, j)$ can be formed by taking the product of the two terms in the expression for $W(i, j)$ instead of the sum, but this has not been experimented with.

In [20], Shahidi and Moloney argued that after $W(i, j)$ has been found for all image pixels, each value should be multiplied by a uniformly distributed random number between 0 and 1. This step is not necessary if random number generation is not available. Instead, as stated earlier, the pixel with the largest $W(i, j)$ can be chosen. In some experiments, the use of random multiplication of $W(i, j)$ and

simply taking the largest $W(i,j)$ without multiplication are compared. In terms of efficiency, it is better not to use the random multiplication step since if we do use it, the possibility of an optimization of *gaps* using a balanced binary tree is eliminated. If the W value of each pixel were to be multiplied at each iteration by a random number, then it would be impossible to store the calculated values from step to step. However, if the random multiplication is not used, then the pixel with the greatest resulting W is chosen as the next sample. There will be more details on this later in Section 3.4.2.

As with nonadaptive *gaps*, only the relative magnitudes of the weights are important, so we can eliminate one of the parameters in Equation 3.3 by normalization, setting M_G to $\frac{M_G}{M_G+M_E}$ and M_E to $\frac{M_E}{M_G+M_E} = 1 - M_G$.

Quasi-adaptive stage In the previous subsection, the formula for the weight function for iterations greater than *cutoff* was given. Now we give the weight function for the initial quasi-adaptive stage. This weight function is more complex than that for the adaptive stage and has the form:

$$\begin{aligned} W(i,j) = {} & M_G \left(\frac{h_{Mask}(i,j)}{\max_{(i',j')\in I} h_{Mask}(i',j')}, l \right) \times F_l \left(\frac{G(i,j)}{\max_{(i',j')\in I} G(i',j')} \right) \\ & + M_E \times E \left(\frac{h_{Mask}(i,j)}{\max_{(i',j')\in I} h_{Mask}(i',j')}, ERank(i,j) \right) \\ & + B \left(i, j, l, \frac{G(i,j)}{\max_{(i',j')\in I} G(i',j')}, \frac{h_{Mask}(i,j)}{\max_{(i',j')\in I} h_{Mask}(i',j')}, I(i,j) \right). \end{aligned} \tag{3.6}$$

We want to sample those pixels with lower h_{Mask} values more frequently in this stage since they will not be sampled as often later on. It may be argued that it would be better to sample those pixels with low Sob values instead, but this does not turn out to be a good strategy for masks which contain pixels more than distance 1 away from their centres, because such pixels will have both high h_{Mask} and low Sob values which means that they will be too heavily favoured. Though low h_{Mask} pixels are preferred in the initial stages, the E term remains for these iterations, and this term is high for large h_{Mask} and $ERank$ values, meaning that the pixels with low Sob and high h_{Mask} values will have a much greater chance of being sampled.

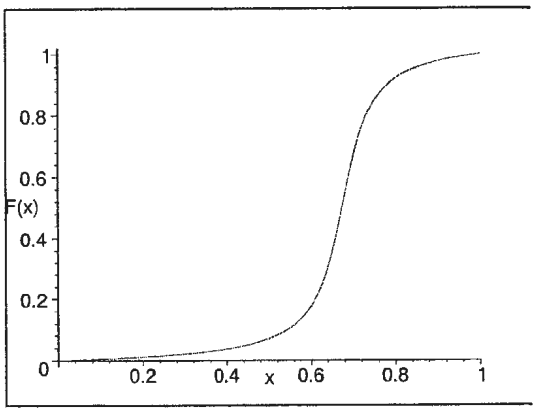


Figure 3.5: Function $F_{\frac{cutoff}{2}}(x)$ used for gap magnitude contribution to pixel weight at iteration $\frac{cutoff}{2}$

For the quasi-adaptive stage, the weight has three terms which are added together as in Equation 3.6. One (the second term), exclusively involves the edge magnitudes of the image, and is identical to the corresponding term in the adaptive stage. In the adaptive stage weight computations, another term (the first) uses the gap magnitudes

$G(i,j)$ of the image with a fixed weight M_G for this term. Before the cutoff, the coefficient M_G of this term is gradually decreased to its final value at the adaptive stage, with this decrease dependent not only on iteration number but also on the h_{Mask} value of the pixel. The higher h_{Mask} is, the lower the weight should be. As well, the function F_l is an arctan curve shifted from right to left. This is shown in Figure 3.5, and is done to increase the likelihood of large gaps over smaller gaps being chosen at the beginning. So, instead of the term $M_G \times F(\frac{G(i,j)}{\max_{(i',j')\in I} G(i',j')})$ where M_G is a constant, and F does not depend on iteration number, as used in the adaptive stage (see Equation 3.3), the more complex formula:

$$M_G\left(\frac{h_{Mask}(i,\,j)}{\max_{(i',j')\in I} h_{Mask}(i',\,j')}\,l\right) \times F_l\left(\frac{G(i,\,j)}{\max_{(i',j')\in I} G(i',\,j')}\right) \tag{3.7}$$

is used in the quasi-adaptive stage.

A smooth transition from quasi-adaptive to adaptive sampling is preferred, so that there is no sudden jump between the two states. Let $C(l) = (\frac{l}{cutoff})^2$. Then, to satisfy all the desired criteria, the functions for this term (Equation 3.7) are defined as follows:

$$M_G(h,l) = 7.0 - 4.3C(l) - 2.0h(1 - C(l)) \tag{3.8}$$

$$F_l(x) = \frac{\arctan(20x - (17 - 14C(l))) + \arctan(17 - 14C(l))}{\arctan(3 + 14C(l)) + \arctan(17 - 14C(l))}. \tag{3.9}$$

The final term $B(i,j,l,\{G(i',j')\}_{(i',j')\in I},\{h_{Mask}(i',j')\}_{(i',j')\in I},I(i,j))$ in the weighting function before iteration *cutoff* tries to increase the weight for pixels in bright homogeneous areas, but also ensures that the distance to previous samples is not too small. The formula for this term is:

$$B\left(i,\,j,\,l,\,\frac{G(i,\,j)}{\max_{(i',\,j')\in I} G(i',\,j')},\,\frac{h_{Mask}(i,\,j)}{\max_{(i',\,j')\in I} h_{Mask}(i',\,j')},\,I(i,\,j)\right) = \tag{3.10}$$
$$2.0\sqrt[3]{\frac{G(i,\,j)}{\max_{(i',\,j')\in I} G(i',\,j')}}(1-C(l))\left(1-\frac{h_{Mask}(i,\,j)}{\max_{(i',\,j')\in I} h_{Mask}(i',\,j')}\right)\left(\frac{I(i,\,j)}{M}\right)$$

All the variables in the above equation have been discussed, except for M, which is simply the maximum gray-level of the image. The higher $I(i,j)$ and the lower $h_{Mask}(i,j)$ are, the higher this B term is overall, and this B part of the entire weight decreases to 0 quadratically. The use of a linear decrease instead of $C(l)$ was found to make the sampling switch to full adaptivity too quickly. Finally, the first part of the product is $\sqrt[3]{\frac{G(i,j)}{\max_{(i',j')\in I} G(i',j')}}$, where clearly as G increases, meaning that the pixel is further away from previously selected samples, the higher this part of the product becomes.

Adaptive *Gaps* Parameters As already mentioned, when an image is being sampled with *gaps* or any other algorithm, it is generally true that it is desirable to sample the image fairly regularly at first, not adapting too much to its specific characteristics. Only after this do we want to capture the edge information - this helps

improve the progressive nature of any irregular sampling scheme, so that if the sampling is stopped at any stage, the reconstruction is more faithful to the original. In the *gaps* algorithm, the transition between the two stages is image-dependent.

Multilevel B-Spline Approximation (MBA) is a scattered data interpolation algorithm which is explained in Section 4.3. There, it will be seen that the spline surface for each level is constructed to minimize the sum of squares deviation of the control grid values from zero. It is necessary therefore to sample homogeneous regions of higher intensity more densely than those with low intensity. In order for *gaps* to work well with MBA, the intensity distribution of the pixels with low edge magnitudes in the image is used. Let $U = \{(i,j)|h_{Mask}(i,j) < T\}$, where T is a parameter that is set to 40, since h_{Mask} values below this are quite low, as determined experimentally. Let the image have N pixels and the average intensity of the pixels in U be $\bar{U}$. Then the cutoff between quasi-adaptive and adaptive sampling is determined as follows:

$$cutoff = \lfloor C_m + \frac{K\bar{U}|U|}{N} \rfloor \tag{3.11}$$

where C_m is the minimum possible *cutoff*, and K is a proportionality constant determining how fast *cutoff* varies with respect to $\frac{\bar{U}|U|}{N}$. For our tests, C_m was set to 150 and K was set to 600. The higher that $\bar{U}$ is, the higher the cutoff is. This is what would be expected since as already said, homogeneous bright regions have to be sampled more densely than dark ones, and so if $\bar{U}$ is higher, so should be the

cutoff for quasi-adaptive sampling. Note that some pixels with $h_{Mask} = 0$ will be those exactly on edges, but since the number of such pixels is very low, and the brightnesses of these pixels are very close to those in the regions they enclose, this does not have much of an effect. Additionally, when $|U|$, the cardinality of U, is larger, this means that there are more homogeneous regions to sample, so the cutoff should also be higher.

Now we move on to the explanation of the $ERank$ function. As the name suggests, the $ERank$ function measures the rank of h_{Mask} of its argument among all the h_{Mask} values in the image. Because there are many pixels with very low h_{Mask} values, the approach is taken of putting the h_{Mask} values in different bins, so that $ERank(i, j)$ is not very high for pixels with very low (less than T) h_{Mask} values. Assume $N(i, j)$ is the number of the bin that corresponds to pixel (i, j). Then $N(i, j)$ is determined as follows:

$$N(i,j) = \begin{cases} 0 & \text{if } h_{Mask}(i,j) < T \\ \lfloor \frac{h_{Mask}(i,j)-T}{N_B} \rfloor + 1 & \text{otherwise} \end{cases} \tag{3.12}$$

where N_B is the number of h_{Mask} values in each "bin". "Bins" are used in order to treat pixels with similar h_{Mask} values in the same manner (at least for their edge terms in their gap magnitude calculations). If there are many pixels with h_{Mask} values close to each other, then their ranks will differ widely, while their raw edge magnitudes will not. The value of N_B is set to 10. This value is experimentally a good one, determined over our image test set, because it is not good for N_B to be so

small that it differentiates too much between pixels with similar h_{Mask} values, while at the same time it should not be too coarse. Let $S_k = \{(i,j)|N(i,j) = k\}$. Then $ERank(i,j) = \sum_{k=0}^{N(i,j)-1} |S_k|$, measuring how important $h_{Mask}(i,j)$ is in relation to the entire image. The reason for using the $ERank$ function is that in some images, especially but not exclusively graphical ones, there are pixels whose raw h_{Mask} values are very low, but which are very important (see, for example, the text "jumped over the" in the graphical test image `qbf` later in Figure 3.7(g)). In Chapter 4, it will be demonstrated how using the $ERank$ function improves the quality of reconstructions from *gaps* samples.

Good values for the edge coefficient M_E and the adaptive-stage gap coefficient M_G were found to be $M_E = 4.5$ and $M_G = 2.7$. Other than that, the mask used is circular with radius 2, and $h_{Mask}(i,j)$ is taken to be $Sob^{Max}_{Mask}(i,j) - Sob(i,j)$. Finally, $fillfactor = 1.4$. It should be greater than 1 to be able to find close by samples, but other than this should be arbitrary. This value was not experimented too much with since it was found not to be crucial to the success of *gaps*.

3.2 Faster Farthest Point Sampling (FFPS)

While *gaps* is an approximation algorithm to Farthest Point Sampling, we have discovered that it is possible to exactly emulate the behavior of FPS (up to pixel position discretization errors) with a new algorithm we call Faster Farthest Point

Sampling (FFPS) and to do so at a speed even faster than *gaps*. The description of the non-adaptive algorithm is very simple and follows next, after which the adaptive algorithm is presented, which is only a slight modification of the non-adaptive version, and which is basically the same as adaptive *gaps*.

As with *gaps* and FPS, the first four samples are chosen to be the corners of the image being sampled, and the fifth sample to be positioned randomly. Setting gm^l to be an array at iteration l of the same dimensions as the input image, the array gm^5 is initialized so that $gm^5(i,j)$ is the distance of pixel (i,j) to its closest sample out of the initial five. To speed up the algorithm, a lookup table for the distances is used, since only distances between points with integer coordinates need to be considered. Let l commence as the index of the next sample to be chosen, in this case, 6. Then the following steps are repeated until the required number of samples have been chosen:

1. Let $r_l = \max_{(i,j)\in I} gm^{l-1}(i,j)$.

2. Let $s_l = (i_l, j_l) = \arg\max_{(i,j)\in I} gm^{l-1}(i,j)$. This is the next sample.

3. Let $gm^l(i,j) = gm^{l-1}(i,j)$ for all i,j.

4. For all $(d_i, d_j) \in D_{r_l} = \{(i,j) \in \mathbb{Z} | i^2 + j^2 \leq r_l^2\}$, update gm^l as follows: If the Euclidean norm of (d_i, d_j) is denoted by $|(d_i, d_j)|$, then let $gm^l(i_l + d_i, j_l + d_j) = \min(|(d_i, d_j)|, gm^{l-1}(i_l + d_i, j_l + d_j))$.

5. Let $l = l + 1$.

Unlike FPS, the efficiency of this algorithm does not go down as the number of samples increases. In fact, the efficiency improves. An example of the addition of a sample and the updating of the gm at the lth iteration is shown in Figure 3.6. As is demonstrated by the diagram, only the gm values of those pixels within $\max_{(i,j)\in I} gm^{l-1}(i,j)$ of the new sample (inside the circle) have to be recalculated.

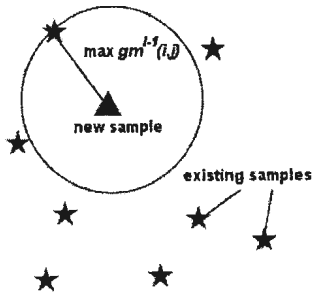


Figure 3.6: Updating of gm^l in FFPS

Theorem

After any iteration l of the above algorithm, $gm^l(i,j)$ always contains the distance of the pixel (i,j) to the closest sample already chosen from $\{s_i\}_{i=1}^{l}$.

Proof (by induction)

Let l be the iteration number, starting at 5. Clearly for $l = 5$, gm^l has the required property by its definition. Assume that the property holds for $l = n$. Let $d^l_{min}(i,j)$ be the minimum distance from (i,j) to any of the first l samples.

It suffices to show that at iteration $l = n+1$, $gm^l(i,j) = d^l_{min}(i,j)$. Obviously, $d^{n+1}_{min}(i,j) = \min(d^n_{min}(i,j), d((i,j),s_{n+1})) = \min(gm^n(i,j), d((i,j),s_{n+1}))$, where $d(\mathbf{x}, \mathbf{y})$ is the Euclidean distance between points $\mathbf{x}$ and $\mathbf{y}$. Since $r_{n+1} = \max_{(i,j)\in I} gm^n(i,j)$, $d^{n+1}_{min}(i,j) = gm^n(i,j)$ for all (i,j) such that $d((i,j),s_{n+1}) > r_{n+1}$. All the other (i,j) can be labelled as $(i_{n+1}+d_i, j_{n+1}+d_j)$ where $|(d_i,d_j)| \leq r_{n+1}$. For these pixels, $d^{n+1}_{min}(i_{n+1}+d_i, j_{n+1}+d_j) = \min(gm^n(i,j), d((i_{n+1}+d_i, j_{n+1}+d_j),(i_{n+1},j_{n+1}))) = \min(gm^n(i,j), |(d_i,d_j)|)$. Since the construction of $d^{n+1}_{min}(i,j)$ is identical to that of $gm^{n+1}(i,j)$, they are identical for all (i,j), and thus the result follows. □

At least one idea exists to make FFPS adaptive. Instead of choosing s_{n+1} as the pixel with the maximum gm^n, it is also possible to consider the Sobel edge magnitudes $Sob(i,j)$, and pick the sample with the maximum weighted combination of gm^n and h_{Mask}, using the same parameters as *gaps*, including the time varying M_G and $F(x)$ for the quasi-adaptive stage. Unfortunately, the functions and parameters for adaptive *gaps* do not work directly with adaptive FFPS, so more experimentation is needed in order to find suitable functions to make this idea work. One advantage of adaptive *gaps* and FFPS over skewness based sampling and adaptive FPS is that instead of relying on local skewness or bandwidth, the adaptation depends solely on the Sobel edge magnitudes. These magnitudes can also be generalized to color images, see for example [21], whereas it is not immediately clear how to do this with

the other measures.

Theoretically, non-adaptive FFPS and FPS have the same order of complexity, namely $O(n \log n)$ because they both use balanced binary trees. However, timing tests between the two show that FFPS is more efficient than FPS for the same number of samples. The high efficiency and simplicity of FFPS as opposed to FPS also means that it is better suited to a hardware implementation.

We can find the complexity of the main part of non-adaptive FFPS (not including the maintenance of balanced binary trees) using the following argument. Suppose that the image being sampled has dimensions $r \times c$ pixels. Then the *Principal Wavelength* [22], or average distance between samples, after k samples have been chosen is approximately $\lambda_k = \frac{1}{\sqrt{\frac{k}{rc}}}$. In [14], it is proven that the maximum ratio between the maximum and minimum distances between samples in a non-adaptive FPS sampling pattern is 2. So the maximum distance between samples after k samples have been chosen is at most $\lambda_k^M = 2\lambda_k$. Thus, for the kth sample that is chosen, we have to update approximately $\pi(\lambda_k^M)^2$ elements of gm. Integrating from 1 to n means that overall, the main part of the algorithm requires about $4\pi rc \ln n = O(\log n)$ operations. In Section 3.4.3, we experimentally compare the running times of the nonadaptive versions of the two algorithms.

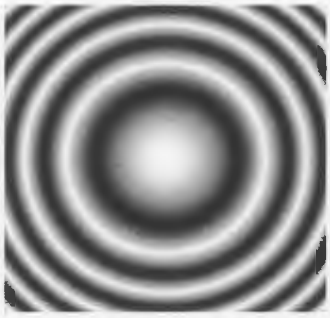
3.3 Test Set

To test the efficiency and the performance of the various sampling, reconstruction and halftoning algorithms described in this thesis, a test set of 8 gray-scale images as shown in Figure 3.7 was used. The names of these images are `chirp`, `antarctica`, `cameraman`, `femme`, `mri`, `peppers`, `qbf`, and `zelda`. The chirp image is a sampling of a mathematical function with formula: $I(i,j) = 0.5\cos(\frac{1}{1000}((i-127.5)^2 + (j-127.5)^2)) + 0.5$ where the image intensities have the range [0,1], and the size of the image is 256x256 pixels ($0 \leq i,j \leq 255$). All the remaining test images were obtained from various sites on the World Wide Web. `mri` is a medical Magnetic Resonance Image of the head, `antarctica` a digital elevation model (DEM) of the Antarctic continent, and `qbf` a typical graphical image. The remaining pictures are natural photographic images. The images are all 256x256 pixels except for `peppers`, which is 255x255 and `antarctica` which is 209x250.

3.4 Test Results

3.4.1 Arrangement of Samples

On page 68, in Figure 3.8, the irregular sampling grids for `zelda` are given, with 3035 samples taken from the adaptive version of each algorithm. The samples are shown as white dots overlaid on the original `zelda` image. As expected, since the sampling



(a) `chirp`

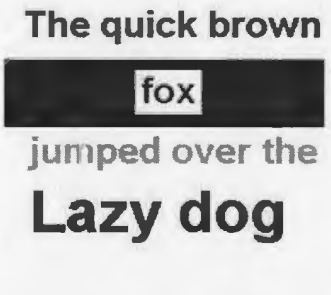
(b) `antarctica`

(c) `cameraman`

(d) `femme`

(e) `mri`

(f) `peppers`



(g) `qbf`

(h) `zelda`

Figure 3.7: Test Set of 8 gray-scale images

algorithms are all adaptive, there are more samples closer to image contours than in uniform regions. It is interesting to compare the constellations for all of the algorithms - it is apparent that there is less clustering to image details with Farthest Point Sampling and unoptimized *gaps* as opposed to the other algorithms. This clustering refers to the fact that for a given edge, there are less samples close to the edge in FPS and unoptimized *gaps* than for the other two sampling algorithms, and there are more samples in regions with lower edge magnitudes with FPS and unoptimized *gaps*. However for FPS, the configuration is a lot more jittered and irregular, especially as compared to the skewness-based samples, which are arranged with double rows of samples along contours. *Gaps* also has this same double-row property to some extent, except that the row on each side is not at a fixed distance away from the edge. This is because a constant radius of exclusion is used for skewness-based sampling, but for *gaps*, samples can be located anywhere in the mask not exactly on the edge. We include the results for both non-optimized *gaps*, where pixel weights are each multiplied by a uniformly distributed random number, and optimized *gaps*, where this multiplication is not performed. For `zelda`, there is clustering evident in the samples taken with optimized *gaps*, this is because adaptivity increases as the algorithm proceeds, but we will see this does not have a significant impact on the reconstruction from these samples. This did not occur with the other test images, since such a large number of samples (3035) was not taken.

(a) 3035 samples taken from `zelda` using FPS

(b) 3035 samples taken from `zelda` using skewness-based sampling

(c) 3035 samples taken from `zelda` using unoptimized *gaps*

(d) 3035 samples taken from `zelda` using optimized *gaps*

Figure 3.8: Irregular sampling grids for different algorithms

Skewness-based sampling takes very few samples in uniform regions - this means that the reconstructions in these areas may not be as good as those from FPS and *gaps*. The quality of reconstructions from this image and the other images in the test set will be compared in the following chapter.

The number of samples that will be taken with skewness-based sampling cannot be predetermined given a set of skewness thresholds and radii of exclusion, whereas the other sampling methods are truly progressive, which means that they can be stopped at any number of samples. So first some good parameters for `zelda` (or any image) with skewness-based sampling are selected and then it is seen how many samples are taken using these parameters. The sampling using the other two algorithms (FPS and *gaps*) is stopped at this same number. The θ_s and r parameters for the skewness-based sampling of the test images are given in Table 3.1. The parameters used for the `femme` and `peppers` images are the same as those in [11], except that the skewness thresholds are smaller by a factor of 100. This is due to an error in [11] as confirmed by correspondence with the first author. Also, the θ^g parameter which is a threshold for determining which pixels have high gradient magnitude (and thus are not included in the sampling grid), was not given in [11], but was found to be $\theta^g = 128$ once again by correspondence with the first author.

3.4.2 Experimental Results

All experiments in this chapter and the next were performed on an AMD 1700XP personal computer. Timings obtained for various algorithms are CPU times, as these are more amenable to comparison. These times are given in Table 3.2. Skewness-based sampling times are not given since they are extremely small (almost instantaneous) compared to the other two algorithms. The parameters for skewness-based sampling used are shown in Table 3.1.

Though the *gaps* sampling times are the highest, they are still not unreasonable. As well, there may be opportunity to further optimize the implementation of this algorithm. One optimization was made when the maximum gap magnitude didn't change in an iteration. In this case, the crosses are not modified, and thus the gap magnitudes only have to be recomputed for pixels bordering the cross of the new sample. A balanced binary tree was also used to store the pixel weights. These two optimizations together have led to significant speed-up as evidenced by Table 3.2. Also, the use of simpler functions, especially in the quasi-adaptive stage would lead to a further improvement in efficiency. In fact, most of the time utilized by *gaps* is spent in the quasi-adaptive stage; if this could be made faster through the use of simpler functions than the arctan, then it should be possible to make *gaps* as fast as FPS.

Image	Skewness θ_s parameters	Skewness r parameters
chirp	[0.0025,0.0063]	15,8,3
antarctica	[0.001,0.005]	15,6,2
cameraman	[0.0025,0.0048,0.0063]	15,8,4,2
femme	[0.00125,0.0038,0.005]	15,11,7,2
mri	[0.00125,0.0038,0.005]	15,11,7,2
peppers	[0.0025,0.0063]	15,8,2
qbf	[0.0025,0.0063]	15,8,2
zelda	[0.0015,0.004,0.008]	15,11,7,2

Table 3.1: Parameters used for skewness-based sampling of test images

Image	No. of Samples	FPS (opt.) time (s)	*gaps* time (s)	*gaps* (part. opt.) time (s)
chirp	2611	8.63	119.93	30.73
antarctica	1439	4.13	65.31	16.63
cameraman	2064	6.47	96.82	28.89
femme	2108	6.26	97.95	28.32
mri	1874	5.55	85.75	21.49
peppers	2523	8.13	112.74	35.20
qbf	2612	8.23	125.20	38.65
zelda	3035	10.79	140.51	35.35

Table 3.2: CPU times for irregular sampling algorithms

3.4.3 FPS vs. FFPS

As mentioned in Section 3.2, both FPS and FFPS have the same order of complexity, namely $O(n \log n)$. Figure 3.9 compares the running times of FPS and FFPS for 500, 1000, 2000 and 3000 samples. Clearly, FFPS is quicker, and since the configurations of samples are the same up to rounding error, they will both have the same minimum distance properties. As a result, nonadaptive FFPS can be used for many applications where such distributions are needed, e.g. positioning of plants within populations and stippling [23]. In Chapter 6, a generalization of FFPS will be applied to halftoning.

Unfortunately, adaptive FFPS, either by using the closest distance to a sample as the gap magnitude in *gaps*, or using the same formula as in [14] for the weighted distance of each pixel does not yield sharp reconstructions. For image sampling, the subpixel resolution becomes more important, and the rounding in adaptive FFPS is too damaging. It should be possible to fix this however, with the addition of random deviations from the integer lattice for example.

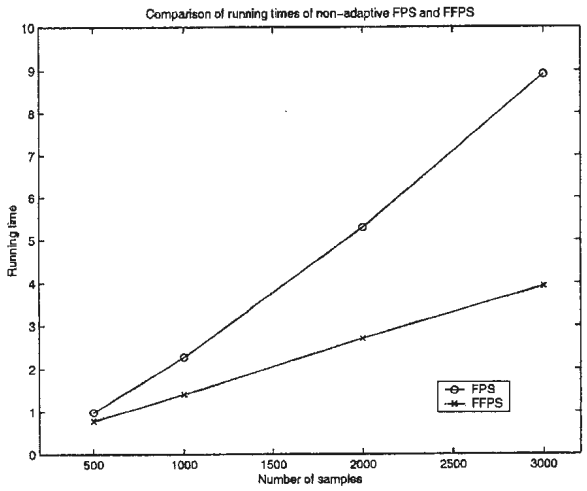


Figure 3.9: Timing Comparison between FPS and FFPS

3.5 Conclusions

It is reported in [11] that the Farthest Point Sampling algorithm in [14] has high complexity. In its optimized form, it was found that this is not the case. Ramponi and Carrato's skewness-based sampling addressed this perceived problem of speed, but unfortunately, parameters must be chosen specific to the images, and it is not

always possible to do this without *a priori* knowledge of the image characteristics. Also, the number of samples we wish to take cannot be chosen beforehand, as this depends on the chosen parameters, and the relationship between the parameters and the number of samples cannot be predicted. The *gaps* sampling algorithm was created to address both the problems of Farthest Point Sampling and Ramponi's skewness-based sampling. Many combinations of parameters were not tested for *gaps* - it is very well possible that others may lead to even better results.

It may be possible to combine the fast time of optimized *gaps* with the slightly improved quality of unoptimized *gaps*. This could be done by simulating the multiplication of the pixel weights by uniformly distributed random numbers, by not always selecting the pixel with the highest non-randomized pixel weight, but instead one with a lower such weight. Another possibility is to use FFPS instead of *gaps* to estimate the distance of a pixel to the closest sample already chosen. Using the same functions and parameters as *gaps* and replacing the gap magnitude by the FFPS distance to the closest sample does not give good results, but using different functions and parameters should give an efficient companion algorithm to *gaps*.

Chapter 4

Existing Scattered Data Interpolation Algorithms

4.1 Introduction

The problem of irregular sampling has already been discussed in the previous chapters. Now the recovery (in these cases lossy) of an irregularly sampled image from its samples is considered. Suppose samples $(x_i, y_i, z_i), 1 \leq i \leq N_s$ have been taken, where $z_i = I(x_i, y_i)$ is the image intensity at pixel (x_i, y_i). Then the problem of scattered data interpolation becomes one of extending the information known on the sampled pixels to all pixels in the image. The interpolated image $I_r(x, y)$ may not even exactly match the data on the samples, so that $I_r(x_i, y_i)$ is not necessarily equal to $z_i = I(x_i, y_i)$ for all $i, 1 \leq i \leq N_s$. Thus calling the recovery process interpolation may be somewhat of a misnomer, and instead the term scattered data approximation

should be used. But because scattered data interpolation is used so widely in the literature, we exclusively use this phrase.

There are many applications of scattered data interpolation, for example image coding, map making, and computer vision [8]. To achieve the goal of reconstructing an image from its samples, many techniques have been proposed. These reconstruction methods can be classified as either local or global. A reconstruction algorithm is local if the addition of a sample only changes the reconstruction in a small region around the added point, whereas it is global if the entire reconstruction changes.

In this thesis, two existing reconstruction techniques pertinent to irregular sampling are looked at:

1. Adapted Nearest Neighbor Interpolation (Adapted 4-NNI)

2. Multilevel B-Spline Approximation (MBA) reconstruction

The former is based on an inverse-distance weighting of the scattered data to the pixels to be interpolated, but is relatively slow, has to be recomputed upon the insertion of any new samples, and produces some artifacts, for example flat areas at sample positions and the appearance of level contours. MBA treats the interpolation of the image as a surface-fitting problem, and is very quick, but does not work well with certain distributions of samples. Adapted 4-NNI is essentially local because an added point will only be one of the four nearest neighbors of the pixel being interpolated for close-by pixels, and so only these have to be updated. However,

if a sample is added in a rather sparsely sampled region, then a large part of the reconstruction has to be changed. MBA is theoretically a global reconstruction algorithm since at the coarsest level, an added sample will influence the values of all control grid points, though since this influence will be very small, we can consider it to be local by only looking at the surface for finer levels. This will become more clear after both reconstruction methods are further described in the next two sections.

4.2 Adapted 4-Nearest Neighbor Interpolation (Adapted 4-NNI)

Four-Nearest Neighbor Interpolation was used in [14] to compare the quality of irregular sampling techniques. This is a very basic interpolation algorithm first put forward by Shepard in 1968 [24], where for any pixel to be interpolated in the reconstructed image, its four nearest neighbors among the samples of known intensity are taken, and the interpolated value is set to be a weighted sum of the four intensities of these nearest neighbors. The weight for each neighbor is inversely proportional to the distance from that neighbor to the pixel being interpolated.

Mathematically,

$$I(i,j) = \frac{1}{\sum_{k=1}^{4} w_k(i,j)} \sum_{k=1}^{4} w_k(i,j) I(i_k, j_k). \tag{4.1}$$

Here $I(i,j)$ is the image value at pixel (i,j), $\{(i_k,j_k), 1 \leq k \leq 4\}$ is the set of four nearest neighbors to (i,j), and

$$w_k(i,j) = \frac{1}{d_k(i,j)} = \frac{1}{d((i,j),(i_k,j_k))}, \tag{4.2}$$

where $d(\mathbf{x},\mathbf{y})$ is the Euclidean distance between pixels $\mathbf{x}$ and $\mathbf{y}$. This reconstruction algorithm is slow for irregular samples compared to MBA because for each pixel in the image, the four nearest neighbors have to be found by searching exhaustively through all the samples. This can be speeded up by using the Voronoi diagram of the samples [14, 25] but this adds significant implementation complexity to the algorithm. There is also another issue related to the fact that if we have a contour with one side bright and the other side darker, and two samples on each side of the edge, but close to each other, then pixels close to the edge and close to the samples, on the bright side of the edge, will be noticably darker than they should be, because the samples on the dark side of the edge will have undue emphasis for these pixels. Ramponi and Carrato [11] recognized this problem and suggested an improved algorithm called adapted 4-Nearest Neighbor Interpolation. The results of adapted 4-NNI as compared to regular 4-NNI are shown in Figure 4.1, taken and modified from [11]. In Figure 4.1(a), the sample positions are marked by dark squares, and the image being sampled is not shown. There is a double row of samples - the row closest to the lower right corner is sampling a brighter regions than the other samples in the

image. The reconstruction using pure 4-Nearest Neighbor Interpolation is shown in Figure 4.1(b), and we see here that the reconstructed edge is very spread out and not distinct. This is as opposed to the adapted 4-Nearest Neighbor Interpolation from the samples in Figure 4.1(a), where the edge is much sharper.

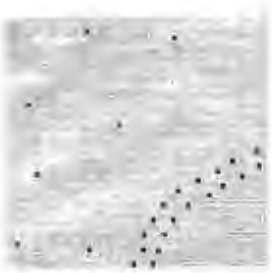
In this adapted algorithm, the formula used to generate the value for a pixel is based on whether it is:

1. Exactly on an edge.

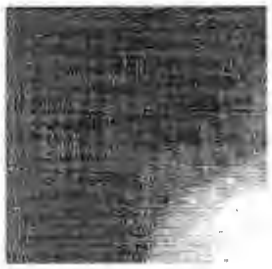
2. Close to an edge, or

3. Far away from any edges.

Let the pixel whose intensity is being interpolated be called p. Then first, it must be detected which type of pixel p is, in relation to the edges of the image. To do this, suppose that the maximum distance from any of the four nearest neighbors of p to p itself is $d_{max}(p)$. Let the maximum distance between any pair of the four nearest neighbors of p be $\hat{d}_{max}(p)$. A threshold $\theta_d > 2r_n$ is selected, where r_n is the radius of exclusion for the highest level n in skewness-based sampling. Then, we consider p to lie exactly on an edge if $d_{max}(p) < \theta_d$; p is identified as being close to an edge if the previous inequality is found not to hold and the inequality $\hat{d}_{max}(p) < \theta_d$ is satisfied; finally, p is far from any edges if both $d_{max}(p) \geq \theta_d$ and $\hat{d}_{max}(p) \geq \theta_d$. Observe that there is no dependence on the sample intensities, just their positions, for determining the position of the pixel to be interpolated with respect to the edges of the image.



(a) Zoom on example samples on contour



(b) 4-NNI reconstruction

(c) Adapted 4-NNI reconstruction

Figure 4.1: Adapted 4-NNI as opposed to 4-NNI reconstruction. (a) Zoom on example samples on contour, (b) 4-NNI reconstruction, and (c) Adapted 4-NNI reconstruction.

It is now examined how the position of a pixel with respect to edges influences in which way the pixel is interpolated.

If the pixel p is exactly on an edge, then only samples on the same edge should

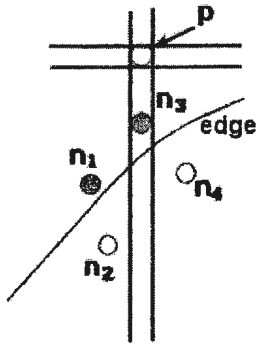


Figure 4.2: Interpolation of pixel p close to edge

be counted, so the weight in Equation 4.2 becomes $w_k(i,j) = \frac{1}{(d_k(i,j))^3}$. This creates a stronger dependence on the samples closer to p. If p is far away from any edges, then the normal 4-NNI determination of the pixel's intensity is adequate. Finally, if p is close to an edge, the characteristics of the skewness-based sampling pattern are used to mainly take into consideration only the samples on the correct side of the edge. It is observed that for every sample on the wrong side of the edge, there is another sample on the right side of the edge which is closer to the pixel begin interpolated, and at a similar angle to the pixel being interpolated. This is used to weight the samples further away less than they normally would be (the weights are divided by a parameter f).

Assuming a pixel p close to an edge is being interpolated, it is necessary to detect when there is a pixel on the wrong side of the edge and weight it less appropriately. This is done by splitting the area around p, as shown in Figure 4.2, into 8 rectangular sectors. The sectors directly bordering p (horizontally or vertically) are only 1 pixel

wide. Then if there are two samples in either the same sector or two adjacent sectors such that the absolute value of the difference of their intensities is at least some threshold θ_v, then the weight of the further pixel is divided by another parameter f. Though it is not specified in [11], a weight for a sample is divided by f at most once.

For the non-image-specific parameters for adapted 4-NNI, the same ones given in the paper by Ramponi and Carrato [11] were chosen, namely $\theta_d = 6$, $\theta_v = 40$ and $f = 50$.

4.3 Multilevel B-Spline Approximation

4.3.1 Introduction

Multilevel B-Spline Approximation (MBA) was published as an algorithm in 1997 [7]. before this reconstruction method came into existence, there were many other algorithms in the public domain which attempted to perform scattered data interpolation, with varying degrees of success. According to Lee *et. al.* [7], there were however diverse problems with the previously existing methods, for example limitations on the distribution of the scattered data, or very high complexity. MBA was designed to overcome these limitations. As will be seen at the end of this section, Multilevel B-Spline Approximation was not found to be as flexible as proposed in [7].

4.3.2 Overview

In [7], a multilevel B-spline interpolator for scattered data called Multilevel B-Spline Approximation (MBA) was introduced. Each level consists of a uniform bicubic B-spline surface defined on a control grid. The control grid at each level has points located on a regular grid, subsampling the image pixel grid. A coarse-to-fine hierarchy is formed starting from the coarsest and lowest level and ending at the finest and highest level.

The further apart the control grid points are, the more slowly changing the spline surface is at that level. The lowest levels are used to fit the broad changes in the scattered data while the higher levels fit to more rapid changes. The final surface that is created to approximate the data is the sum of the surfaces at all the levels. The control grid values are determined by the samples which are close-by using a least-squares criterion. The algorithm starts from the lower levels and works its way up, at each iteration subtracting the surface value at the samples to create residual sample values which are fitted by the higher levels. More details about the algorithm are found in the next subsection.

4.3.3 The algorithm

As already mentioned, MBA treats the problem of scattered data interpolation as one of surface approximation, consisting of surfaces at many levels which are summed.

Each level l, $0 \leq l \leq N$, is a uniform bicubic B-spline surface on an $m_l \times n_l$ control grid Φ_l. The control grid for each level (except possibly for the finest level) is twice as dense in each direction as the control grid for the previous level. Thus, for an image to be sampled and reconstructed of size $r \times c$ pixels, $m_0 = n_0 = 4$ and $m_l = 2m_{l-1} - 1$, $n_l = 2n_{l-1} - 1$, $1 \leq l \leq N-1$ and $m_N = r + 3$ and $n_N = c + 3$.

For the control grid at level l, i.e. Φ_l, ϕ_{mn} located at $(\phi^x_{mn}, \phi^y_{mn})$ is the control point value on the m^{th} column and n^{th} row of the control grid, where the indices m and n start from zero. Suppose the horizontal coordinate in the image is represented by x and the vertical coordinate by y, where the vertical coordinate increases from top to bottom. Let the function $L(x)$ take the value of the horizontal index of the control points directly to the left of x (if x lies exactly at the same horizontal position as a column of control points, then take the index of this column). Similarly, let $U(y)$ be the vertical index of the control points directly above or exactly at y. Then, let $X(m)$ be the x-coordinate of the mth column of the control grid, and let $Y(n)$ be the y-coordinate of the nth row. Finally, let s and t be defined by the following two equations:

$$s = \frac{x - X(L(x))}{X(L(x)+1) - X(L(x))}, \tag{4.3}$$

$$t = \frac{y - Y(U(y))}{Y(U(y)+1) - Y(U(y))}. \tag{4.4}$$

An example of a small grid and the calculation of s and t are shown in Figure

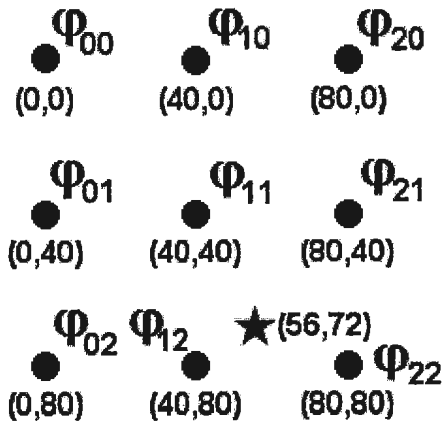


Figure 4.3: Example control grid for Multilevel B-Spline Approximation

4.3. Suppose the point with coordinates $x = 56$ and $y = 72$ is being considered. Then $L(x) = 1$ because column 1 of the control grid is immediately to the left of (56,72). Similarly, $U(y) = 1$, because row 1 is immediately above $y = 72$. $X(L(x)) = X(1) = 40$ and $X(L(x) + 1) = X(2) = 80$. Additionally, $Y(U(y)) = Y(1) = 40$ and $Y(U(y) + 1) = Y(2) = 80$. So $s = \frac{56-40}{80-40} = 0.4$ and $t = \frac{72-40}{80-40} = 0.8$.

Consider the following cubic polynomial functions:

$$B_0(u) = \frac{(1-u)^3}{6}, B_1(u) = \frac{3u^3 - 6u^2 + 4}{6}, B_2(u) = \frac{-3u^3 + 3u^2 + 3u + 1}{6} \text{ and } B_3(u) = \frac{u^3}{6}.$$

Then the spline surface at each level is of the form

$$f(x,y) = \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)\phi_{(i+k)(j+l)}, \tag{4.5}$$

where $i = L(x) - 1$ and $j = U(y) - 1$.

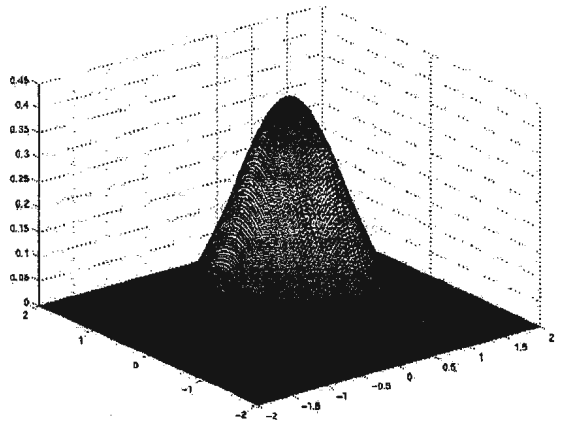


Figure 4.4: Elementary tensor-product B-spline

In Figure 4.4, the resulting spline surface is shown when $\phi_{00} = 1$ and all other ϕ_{ij} are 0. Any arbitrary B-spline surface of the form given in Equation 4.5 with control grid values ϕ_{ij} is a sum of shifted and scaled versions of this elementary tensor-product B-spline. There is a replica at each control grid position, and every replica is scaled by the value of the control grid at the corresponding point.

As will be soon seen, the control point values for the surface at each level are determined so that the sum of squares of errors in the contribution of the control grid values to surrounding sample values is minimized. Each sample can be used to determine the control grid values in its immediate proximity. There are in fact many possible combinations of control grid values around a sample which can interpolate the given sample value, but the spline surface at each level is chosen so that the sum of squares deviation of the control grid values from zero is minimized. This can be achieved either by using the pseudoinverse, or Lagrange multipliers. The

derivation by Lagrange multipliers is used here, as it relies on more elementary and better known mathematics. Suppose that a sample at pixel (x_d, y_d) is given with intensity z_d. For the sake of slightly cleaner algebra, it can be assumed without loss of generality that (x_d, y_d) lies in the upper- left corner of the image, so that it is between $\phi_{11}, \phi_{12}, \phi_{21}$ and ϕ_{22}. In other words, $\phi^x_{11} = \phi^x_{12} \leq x_d < \phi^x_{21} = \phi^x_{22}$ and $\phi^y_{11} = \phi^y_{21} \leq y_d < \phi^y_{12} = \phi^y_{22}$. This means that only the control grid points $\phi_{kl}, 0 \leq k, l \leq 3$ are used to determine the B-spline surface value at (x_d, y_d).

If s and t are defined as above in Equations 4.3 and 4.4, then to successfully interpolate the value z_d at (x_d, y_d), the equation $G(\phi) = \sum_{k=0}^{3} \sum_{l=0}^{3} w_{kl}\phi_{kl} = z_d$ must be satisfied. Consistent with Equation 4.5, $w_{kl} = B_k(s)B_l(t)$. Because the surfaces for each level are added, the deviation of the control grid values from zero for each level's surface is minimized. So $F(\phi) = \sum_{k=0}^{3} \sum_{l=0}^{3} \phi^2_{kl}$ is minimized. In general, if $F(\phi)$ is to be minimized subject to $G(\phi) = c$ where c is a constant, the minimum is chosen as the solution to the system of equations

$$\begin{cases} \nabla F(\phi) = \lambda \nabla G \\ G(\phi) = c. \end{cases} \tag{4.6}$$

In this particular case, $2\phi_{kl} = \lambda w_{kl} \Rightarrow \phi_{kl} = \hat{\lambda} w_{kl}$, where $\hat{\lambda} = \frac{\lambda}{2}$ for the gradient equations, which upon substitution into the constraint equation (the second equation in 4.6) yields $\hat{\lambda} = \frac{z_d}{\sum_{m=0}^{3} \sum_{n=0}^{3} w^2_{mn}} \Rightarrow \phi_{kl} = \frac{w_{kl} z_d}{\sum_{m=0}^{3} \sum_{n=0}^{3} w^2_{mn}}$.

The surface can be made to interpolate all the scattered data if they are sufficiently far apart. This would occur when each control grid value has its value determined by at most one scattered datum point. In most cases, however, especially in lower levels where the control grid points are far apart, the above solution procedure will give more than one different value for some points of the control grid. Let the set of scattered data points be $\{p_d = (x_d, y_d), 1 \leq d \leq N\}$. Supposing that $I_{kl} = \{d | p_d$ gives a value $\phi_d \neq 0$ to $\phi_{kl}\}$, then if I_{kl} is empty, we set ϕ_{kl} to 0. Otherwise, a compromise between the different values assigned to ϕ_{kl} is made. For a given scattered datum point p_d, once again assuming without loss of generality that it is between $\phi_{11}, \phi_{12}, \phi_{21}$ and ϕ_{22}, $z_d = \sum_{m=0}^{3} \sum_{n=0}^{3} \phi_{mn} w_{mn}$, which means that the contribution of ϕ_{kl} to z_d is given by $\phi_{kl} w_{kl}$. Since $\phi_{kl} w_{kl}$ is associated with z_d, label it as $\phi_d w_d$. This is done for all indices in I_{kl}. Then ϕ_{kl} is chosen to minimize the sum of squares error of the contributions to the scattered data values in I_{kl}, this error being $\sum_{d \in I_{kl}} (\phi_{kl} w_d - \phi_d w_d)^2$. The error is minimized when the derivative is equal to zero, which here occurs when $\phi_{kl} = \frac{\sum_{d \in I_{kl}} \phi_d w_d^2}{\sum_{d \in I_{kl}} w_d^2}$.

The value of the surface at each of the samples is subtracted from the sample value, and the residual sample values are used to fit a spline surface at the next level. The approximated image is taken to be the sum of the spline surfaces of all levels. The coarser levels take care of the broad changes in the image, while sharper details are captured by the finer levels.

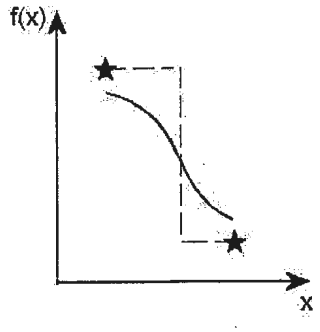


Figure 4.5: Why edges are sometimes blurred with MBA - a 1-D simplification

Unfortunately, for certain sampling distributions, especially those for which samples are not taken exactly on edges, but instead a little further away, MBA can lead to the blurring of edges. Examples of algorithms generating these types of distributions are skewness-based sampling and *gaps*. This is a direct result of the coarse-to-fine hierarchy. In Figure 4.5, the vertical axis represents the intensity of the corresponding position on the horizontal axis. The values of two samples are shown with astericks. As shown in the figure, at a coarser level, since the samples on either side of the edge are rather far apart, it is possible for the more slowly varying spline surface to fit the sample values very well. If the samples were taken around an ideal step edge, shown dashed in Figure 4.5, its reconstruction will be fuzzy. This is because at higher-resolution levels of MBA, the residual values at the samples are very small, so the algorithm will not do much more, and the finer spline levels will not make much of a contribution to the final surface sum. So the broadly changing lower level curve will serve as the major part of the reconstruction of the edge.

(a) (b)

Figure 4.6: Effect of the mask radius on the MBA reconstructions of *gaps*-sampled `zelda`: (a) Radius 2 and (b) Radius 1

This problem with MBA can be remedied for *gaps* by changing the size of the mask used for calculation of the h_{Mask} value. For most of our experiments, a circular mask of radius 2 was used; if instead we use a circular mask of radius 1, then the samples will be closer to edges in the images. An example of doing this for a natural image, `zelda`, is shown in Figure 4.6, where once again 3035 samples with each of the two compared *gaps* algorithms with different circular mask radii. The reconstruction with mask radius 1 is clearly sharper than that for mask radius 2, however this comes at the cost of decreased Signal-to-Noise Ratio (SNR), a quantitative measure of image quality which is described in the next section (the SNR is 12.81 for mask radius 2 vs. 9.70 for mask radius 1). This is due to a lower quality reconstruction of

the background of the image in `zelda`. Since it is not possible to perform a similar adjustment of the skewness-based samples, the reconstruction algorithm is used to improve image quality. This we do in the subsequent chapter.

Now some test results from the various irregular sampling algorithms and the scattered data interpolation techniques presented in this chapter are given.

4.4 Test Results

To compare the quality of different irregular sampling algorithms, images from scattered data produced by different sampling methods must be reconstructed. To make this comparison, it is possible to look at qualitative aspects of the reconstructions, for example the quality of contours and details. But additionally, a quantitative measure can be made to illustrate the calibre of the resulting reconstruction using either the Mean-Squared Error (MSE) or the Signal-to-Noise Ratio (SNR), both of which compare the reconstruction to the original image in a global sense. Let the original image be O and the reconstructed image be R both with r rows and c columns. Let the intensity of the pixel at the i^{th} row and the j^{th} column of R be R_{ij} and of O be O_{ij}. Then the MSE is defined as

$$MSE = \frac{1}{rc}\sum_{i=0}^{r-1}\sum_{j=0}^{c-1}(R_{ij} - O_{ij})^2. \tag{4.7}$$

The SNR (in decibels) is given by the formula [8]:

$$SNR = 10\log_{10}\frac{\sum_{i=0}^{r-1}\sum_{j=0}^{c-1}(O_{ij}-\bar{O})^2}{\sum_{i=0}^{r-1}\sum_{j=0}^{c-1}(R_{ij}-O_{ij})^2} = 10\log_{10}\frac{\sum_{i=0}^{r-1}\sum_{j=0}^{c-1}(O_{ij}-\bar{O})^2}{rc\times MSE}, \quad (4.8)$$

where $\bar{O}$ is the average gray-level of the original image. In this thesis, the SNR is used to evaluate the quality of reconstructions from irregular samples.

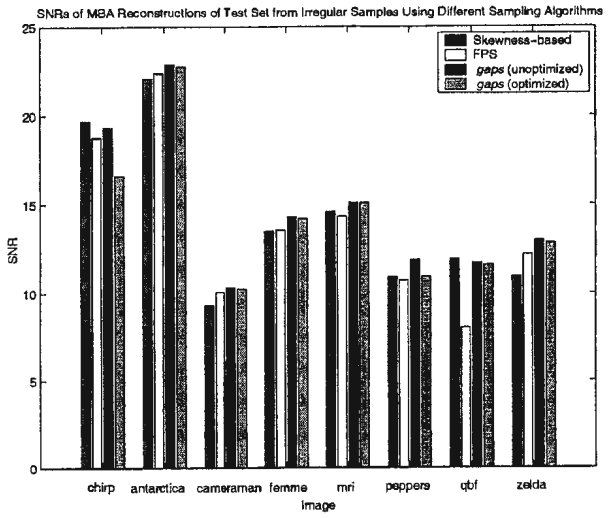
Now the various sampling and reconstruction algorithms are compared. For each of the test images in Figure 3.7, the SNRs of the MBA and adapted 4-NNI reconstructions are computed for the following sampling algorithms: skewness-based sampling, Farthest Point Sampling, and *gaps*. For all sampling algorithms, the same number of samples per image are used (the number determined by skewness-based sampling). Approximate bitrates (bits per pixel) are given in Table 4.1 for all sampled versions of images in the test set. Note that these bitrates, though low, are higher than they could be because no further lossless compression of the sample positions or intensities are taken. When this is done, along with quantization of sample intensities, a bitrate of 0.31 bpp is obtained for the `femme` test image, as reported in [11]. This is even with transmission of the point map which wasn't assumed in Table 4.1. Bar graphs for the SNRs of the reconstructions from these various sampling algorithms are given in Figure 4.7. Observe that the SNRs for *gaps* are slightly less for optimized *gaps* (no multiplication by a random number) as opposed to unoptimized *gaps* (where multiplication by a random number is performed), but this difference is not too great. There

are only two test images for which the *gaps* algorithms are not the best in terms of reconstructed SNR, `chirp` and `qbf`. For `chirp`, the samples from the two *gaps* algorithms give poorer reconstructions than the two existing sampling algorithms, FPS and skewness-based sampling, for both reconstruction methods (MBA and adapted 4-NNI). For `qbf`, only the MBA reconstructions from the *gaps* algorithms are worse than skewness-based sampling but they are still better than FPS. The difference is not substantial for `qbf`, but for mathematical functions like `chirp`, *gaps* should not be used. Since mathematical functions are not however a major class of images, this is not too important, and with more work, this class of images should also be able to be sampled effectively with *gaps*.

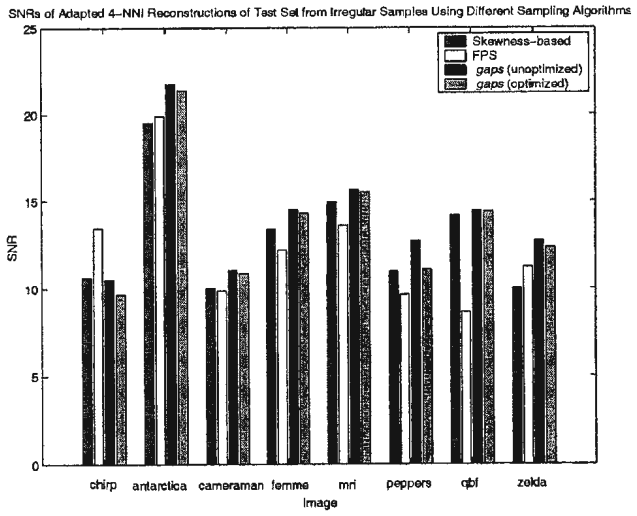
For the adapted 4-NNI reconstructions from *gaps*, the parameter $\theta_d = 10$ was used. The reconstructions of two representative test images, `mri` and `qbf`, are examined more closely. The `mri` image is chosen because it highlights the inability of skewness-based sampling to pick up the less visible but still important details of this image. The `qbf` image is selected because it demonstrates the ineffectiveness of Farthest Point Sampling for graphical images. *Gaps* was found to work better than skewness-based sampling and FPS for natural images as well, as is evidenced by higher reconstructed SNRs. These are shown later in Figure 4.7.

Image	Number of Samples	Approximate Bitrate (bits per pixel)
`chirp`	2611	0.966
`antarctica`	1439	0.661
`cameraman`	2064	0.756
`femme`	2108	0.772
`mri`	1874	0.686
`peppers`	2523	0.931
`qbf`	2612	0.957
`zelda`	3035	1.111

Table 4.1: Bitrates for given number of samples for images in test set, assuming transmission of point map, no quantization of sample intensities, and no lossless compression of image data



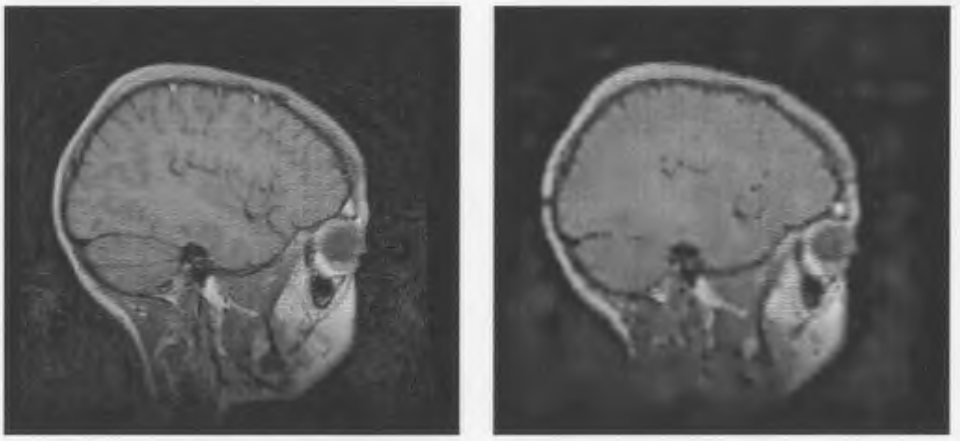
(a) SNRs of MBA reconstructions of test set from irregular samples of different algorithms



(b) SNRs of Adapted 4-NNI reconstructions of test set from irregular samples of different algorithms

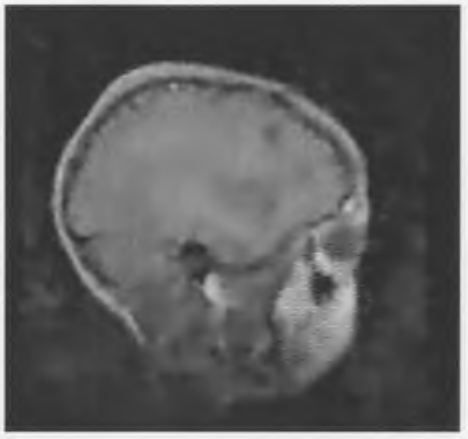
Figure 4.7: Comparison of SNRs of MBA and Adapted 4-NNI reconstructions of test images from skewness-based sampling, Farthest Point Sampling, and *Gaps*

The recovered images from `mri` samples obtained with different irregular sampling algorithms using MBA and adapted 4-NNI are shown in Figures 4.8 and 4.9 respectively. It is seen that in terms of SNR, both optimized and unoptimized *gaps* are better than FPS and skewness-based sampling. This holds for the two reconstruction algorithms, MBA and adapted 4-NNI. One advantage of *gaps* over skewness-based sampling is that in this case, for both reconstruction algorithms, the ridges of the brain are more visible in the reconstructions from *gaps* than the skewness-based samples. There are also two bright dots on the top of the brain in the original `mri` image which can't be seen in the reconstructions from the skewness-based samples, but which are visible in the reconstructions from *gaps*. Farthest Point Sampling gives the worst results for the given number of samples (1874), and this was found to be the case with all images in the test set except for `chirp`, for which *gaps* was found to be inferior to the remaining sampling algorithms. The interface between the brain and the skull is not completely black as it should be, and many of the details on the base of the skull cannot be seen. These types of details are very important especially for medical images, where diagnoses may be made based on this visual information. FPS is especially bad for adapted 4-NN interpolation. The edges are very jagged and not smooth. This is a direct result of the lack of structure in the sampling patterns around edges, and was found to occur with all FPS-sampled test images with adapted 4-NNI. The edges from *gaps* reconstructions with adapted 4-NNI are less smooth than those from skewness-based sampling, but are still acceptable.

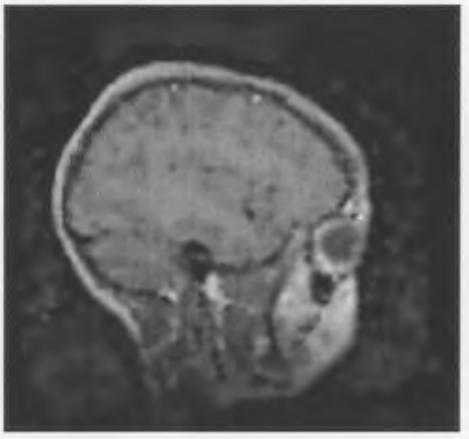


(a)

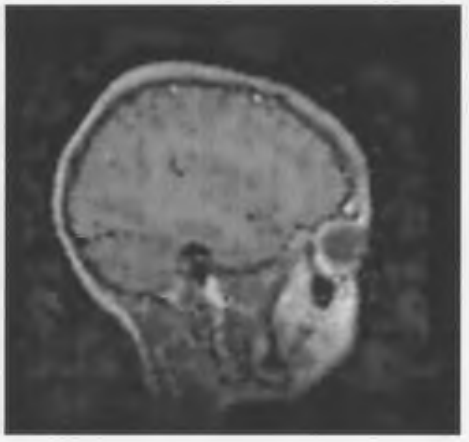
(b)



(c)

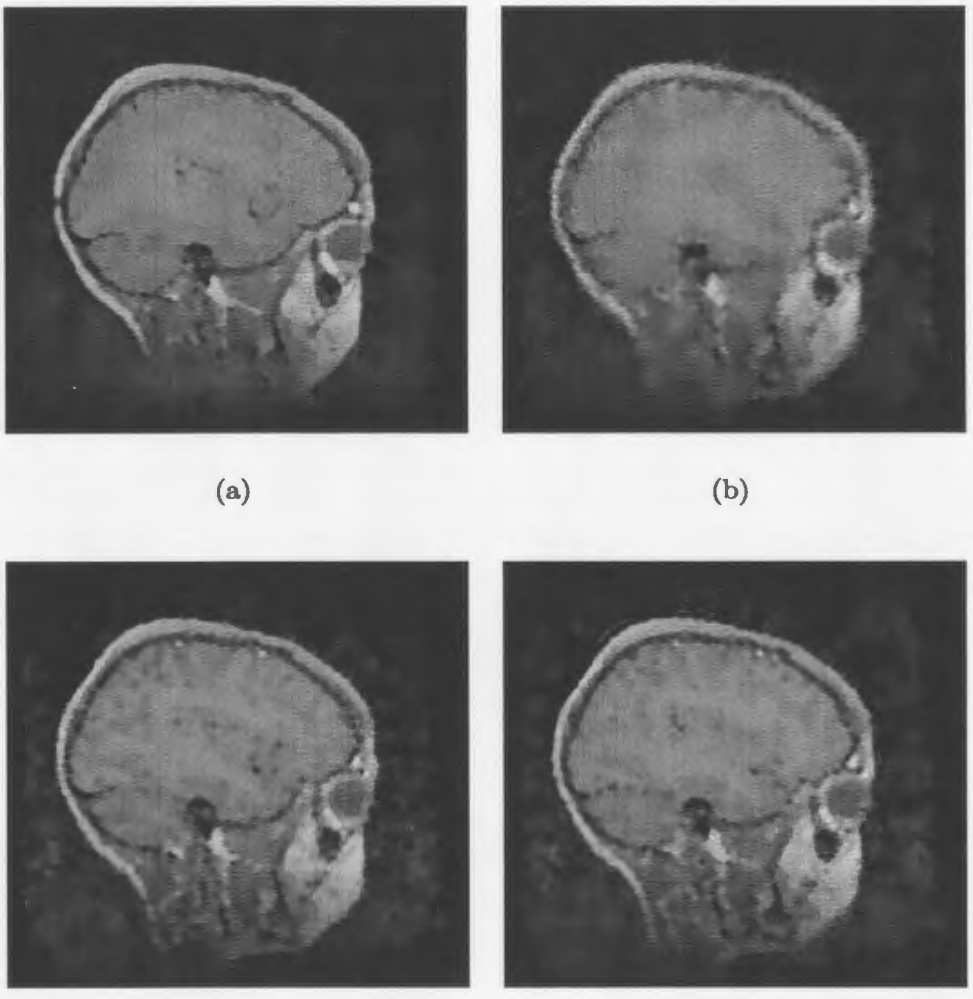


(d)



(e)

Figure 4.8: MBA reconstructions of the `mri` image (a) from (b) Skewness-based sampling, (c) Farthest Point Sampling, (d) Unoptimized *Gaps* and (e) Optimized *Gaps*



(a) (b)

(c) (d)

Figure 4.9: Adapted 4-NNI reconstructions of the `mri` image from (a) Skewness-based sampling, (b) Farthest Point Sampling, (c) Unoptimized *Gaps*, and (d) Optimized *Gaps*

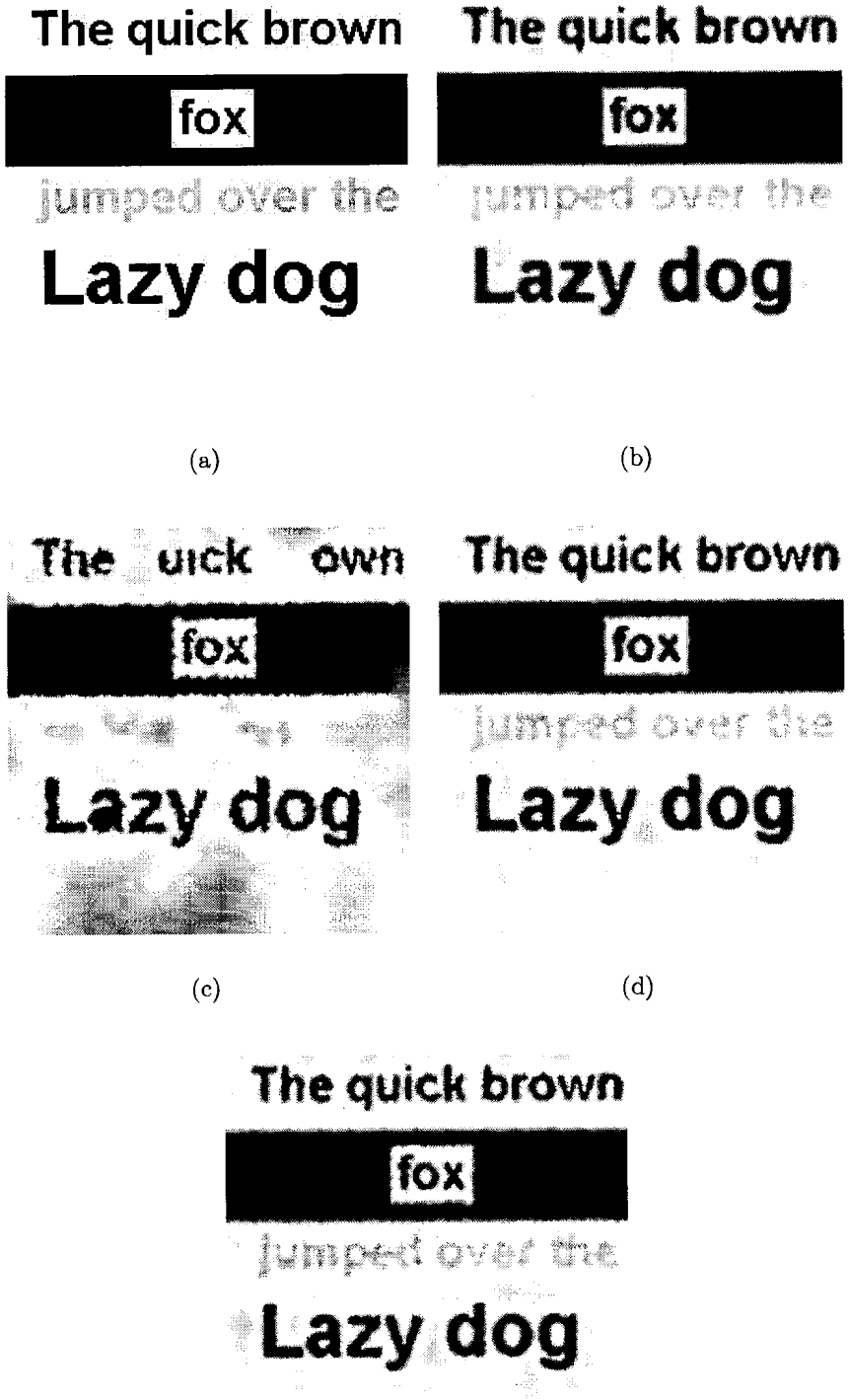


Figure 4.10: MBA reconstructions of the `qbf` image (a) from (b) Skewness-based sampling, (c) Farthest Point Sampling, (d) Unoptimized *Gaps*, and (e) Optimized *Gaps*

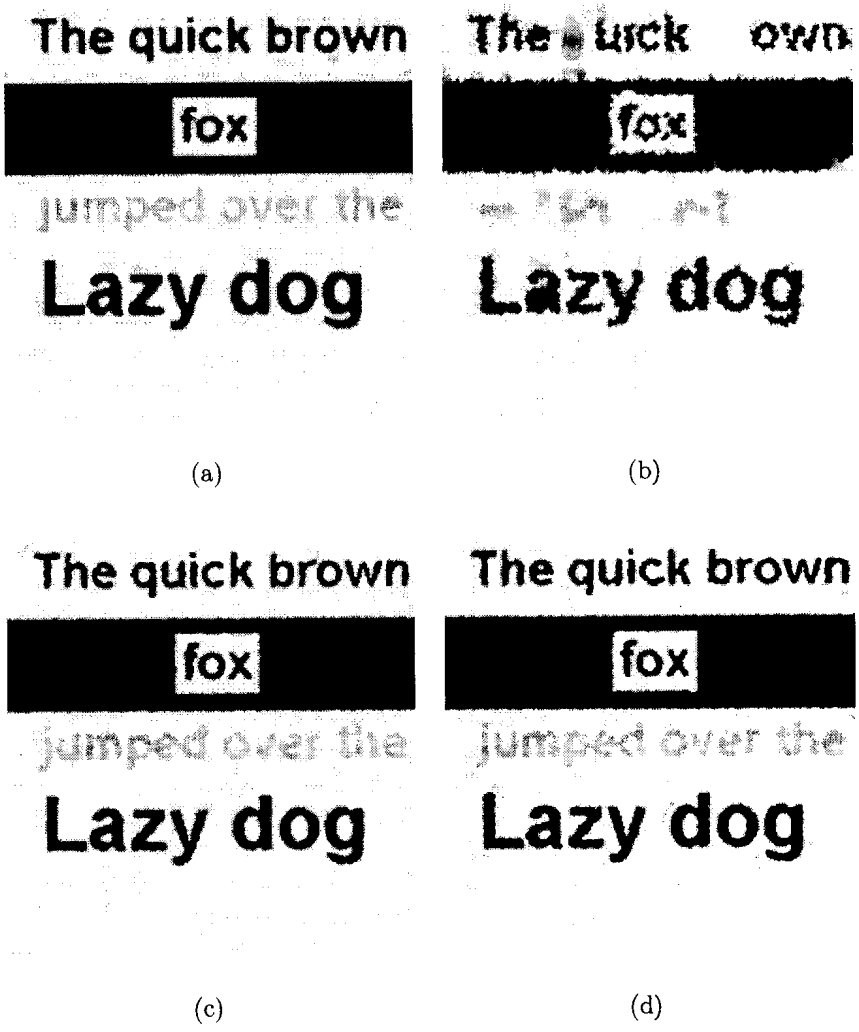


Figure 4.11: Adapted 4-NNI reconstructions of the `qbf` image from (a) Skewness-based sampling, (b) Farthest Point Sampling, (c) Unoptimized *Gaps*, and (d) Optimized *Gaps*

qbf is a typical graphical image, and once again *gaps* outperforms both FPS and skewness-based sampling. The recovered images using Farthest Point samples, skewness-based samples and *gaps* (optimized and unoptimized) samples are shown in Figure 4.10 (for MBA reconstructions) and 4.11 (for adapted 4-NNIs). Using MBA, *gaps* and skewness-based sampled qbfs are about the same - the edges of the MBA reconstruction from skewness-based samples are smoother than those from *gaps* - this is because the skewness-based samples are all at a fixed distance away from the edge, whereas there is more leeway for the placement of *gaps* samples (they are either 1 or 2 pixels away from the edge). On the other hand, the dot of the "j" in "jumped" is visible in the *gaps* reconstructions, but not the skewness-based reconstructions.

There are severe problems with the reconstructions of qbf from Farthest Point samples. This is typical of FPS on most graphical images. The white background is undersampled, leading to many dark parts, due to the already mentioned behavior of MBA, which tries to minimize the deviation from 0 where there are too few samples. The black "q" and "br" are not sampled at all. This is because $N(v)$ for the vertices of the Voronoi diagram on these letters have all three nearest neighbors in the background, so the weight is 0 and these vertices are never chosen as the next sample. The light gray part of the text, "jumped over the", is also severely undersampled. Once again in the adapted 4-NNI reconstruction of qbf from FPS, there are many jagged edges.

4.5 Conclusions

In this chapter, two existing scattered data interpolation algorithms, adapted 4-Nearest Neighbor Interpolation and Multilevel B-Spline Approximation, were described. Tests were performed using these two methods on our test set to compare the different irregular sampling algorithms. These showed that *gaps* had the good properties of FPS, for example a true progressive nature, and like skewness-based sampling, worked across all images (except for `chirp`, which does not represent a major class of images). *Gaps* was superior to skewness-based sampling in that it sampled some parts of the image which had weak edges but were still important to the overall image, more densely than skewness-based sampling. Also image-specific parameters did not have to be chosen.

Chapter 5

New Edge-Directed Multilevel B-Spline Approximation

5.1 Introduction

As mentioned at the end of the previous chapter, MBA has problems with sampling distributions in which samples are relatively far away on both sides of edges. New Edge-Directed Multi-level B-Spline Approximation (NEDMBA) is proposed in this thesis as a way to improve MBA. It is based on MBA, but with improvements adapted from two techniques from the literature, namely an edge-preserving image zoomer (New Edge Directed Interpolation), and a technique called image inpainting. Explanations of both methods used as well as how they fit into the scattered data interpolator called NEDMBA follow. A concise summary of NEDMBA was published in [26].

5.2 New Edge Directed Interpolation

New Edge Directed Interpolation (NEDI) was proposed in [27] as a method to zoom in on a low-resolution image without the blurring artifacts evident in the commonly used bilinear or bicubic interpolation. The core of the NEDI algorithm only doubles the size of an image in each dimension. To zoom in K times on an image, the image size is doubled $\lceil \log_2(K) \rceil$ times, and then downsampled by the appropriate factor. Suppose $X_{i,j}$ is the low resolution $r \times c$ image and $Y_{i,j}$ is the zoomed image of size $2r \times 2c$. Then in the image doubling operation, $Y_{2i,2j}$ is set to equal $X_{i,j}$ for $0 \leq i \leq r-1$, $0 \leq j \leq c-1$. Then it is assumed that $Y_{2i+1,2j+1}$ is a linear combination of the four pixels around it with both indices even (i.e. these are values from the original image). More precisely, suppose

$$Y_{2i+1,2j+1} = \alpha_0 Y_{2i,2j} + \alpha_1 Y_{2i+2,2j} + \alpha_2 Y_{2i+2,2j+2} + \alpha_3 Y_{2i,2j+2} \tag{5.1}$$

for some $\alpha_k, 0 \leq k \leq 3$. An estimate of the optimal MMSE values of the α_k from [27] is:

$$\vec{\alpha} = (C^T C)^{-1} C^T \vec{y}. \tag{5.2}$$

C and $\vec{y}$ are obtained from the intensities in the low-resolution original image in an $M \times M$ neighborhood of the high-resolution pixel being interpolated. $\vec{y}$ is a vector containing the intensities of pixels in the $M \times M$ neighborhood, while the k^{th}

column of C contains the four interpolating neighbors (in counter-clockwise order and starting from the upper-left) of the k^{th} pixel in $\vec{y}$. After the pixels in $Y_{i,j}$ with both indices odd have been successfully interpolated, the pixels with exactly one odd index can be interpolated by repeating the above procedure except that the entire plane is rotated by $\frac{\pi}{4}$ radians. Equation 5.2 is a specific instance of the Wiener-Hopf equation of adaptive filter theory.

In personal communication with Wenmiao Lu, he stated that using an $M \times M$ neighborhood for finding $\vec{\alpha}$ is equivalent to using a weighted-average filter and that this leads to blurring. So instead, only pixels on the same side of the edge of the pixel being interpolated should be used in the calculation of $\vec{\alpha}$ [28]; of course this means that a bigger window should also be used. A variant of this idea was used - instead of using a square window, low-resolution group G of pixels is grown around the pixel being interpolated, containing pixels with high edge magnitude. The constraint $|G| \geq m_G$ is imposed, and the group is not allowed to grow beyond M_G members. Here m_G and M_G are thresholds. If G cannot be grown to at least m_G pixels, then the usual $M \times M$ square window is used. For NEDMBA, the parameter settings $M = 4$, $m_G = 10$ and $M_G = 40$ were used.

5.3 The Main Algorithm - New Edge Directed Multilevel B-Spline Approximation (NEDMBA)

NEDMBA represents a hybrid of two interpolators, Multilevel B-Spline Approximation (MBA) and New Edge Directed Interpolation (NEDI) and incorporates image inpainting for image reparation stages. First the MBA control grid values ϕ_{mn} are generated at each level using MBA. Then the function values at the control grid points are computed by using Equation 4.5 with $s = t = 0$ and the control grid values in a 3x3 neighborhood of the control grid point. Using the function values at the control grid points, the 2x interpolator of NEDI is zoomed as many times as necessary and then downsampled to obtain a surface at pixel resolution. In other words, the function values defined by the spline surface are found at each of the control grid points and instead of using the spline surface to find the function (image intensity) at pixels not exactly located at control grid points, the function at the control grid points is zoomed as much as needed using NEDI to the same size as the image. This is added to the surface sum from the previous levels and this process is continued until the finest level is reached. This finest level does not have to be processed with NEDI at all since the control grid coordinates coincide exactly with the pixels of the image.

As with NEDI, those pixels with both odd indices are first interpolated, and then those with exactly one odd index. Only a generic one of these stages is described for brevity. The variance of the function values in an $M_V \times M_V$ neighborhood of each point to be interpolated is calculated. Then NEDI is used to interpolate those pixels in the top P_V proportion of all the variances for the stage. If a pixel being interpolated doesn't belong to this top proportion, then regular MBA is used to find its value. In NEDMBA, the parameters $M_V = 4$ and $P_V = 0.9$ were chosen for all levels.

Note that the function values and not the control grid values are used for the MBA interpolation - this doesn't affect the resulting intensity too much since this occurs in relatively homogeneous regions, but does result in lower SNRs, as is discussed in the experimental results section.

5.4 Formation of Bright and Dark Spots

NEDMBA using only NEDI and MBA is straightforward to implement but can lead to problems. The major difficulty is the formation of bright and dark spots in the reconstructed image. This is caused by the erroneous localization of an edge, as demonstrated in Figure 5.1 for the case of a bright spot. Because of poor edge localization in one of the levels, samples which are on the bright side of an edge can have a low intensity similar to the dark side of the edge. This means in the next

level of MBA the residual value of these samples will be very high. Then MBA will overcompensate for these high values, and in the next level the very bright values of these samples will be reflected in very bright control grid values in the vicinity which leads to excessive brightness where the cumulative image to that point was already bright as it should have been (since it was on the bright side of the edge). These bright spots persist because the cumulative sample values are close to what they should be starting from the level where the bright spot first formed. The scenario for dark spots is similar.

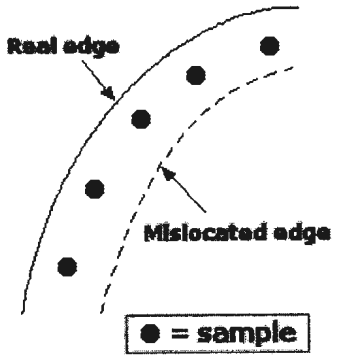


Figure 5.1: Figure showing mislocation of edge leading to formation of bright and dark spots

Instead of NEDI, Weighted Bicubic Spline Interpolation [29], or other potentially more efficient interpolation algorithms can be used. Our multi-level image inpainting framework can be once again brought to bear on the formation of bright and dark spots.

5.5 Selection of Extreme Regions to Fill In

It would be good if the problem of the formation of bright and dark spots could be nipped in the bud so that they would not appear in the first place, however a less direct but potentially mathematically simpler approach is to post-process the image after each level to remove the extreme regions (bright and dark spots). The situation for bright spots is described, and the procedure for dark spots is analogous to this. There are several parameters, e.g. DF_k and P_G which are used in the below discussion; they are clarified by their typical values in Section 5.7.

Now the criteria for finding which parts of the image are extreme bright regions are outlined. These extreme bright regions are the ones that have to be repaired by the inpainting process. Suppose that level k in the hierarchy of lattices in MBA is being considered. One thing to notice is that bright spots usually occur in level k in (connected) bright regions where there is a large spread in the values of the cumulative approximation up to that level. Let $f_k(i,j)$ be the intensity of the pixel in row i and column j in level k and $c_k(i,j) = \sum_{l=0}^{k} f_l(i,j)$. First connected regions are grown of all pixels (i,j) where $f_k(i,j)$ is in the top P_B proportion of all intensities in level k. Region growing is done along the 8 compass directions (north, south, east, west, north-east, north-west, south-east, and south-west). Only regions where there is at least Df_k difference between the maximum and minimum values of f are candidates to be post-processed. Let R be any one of these regions. Note the

dependence on the level k. Let $m_R = \min_{(i,j)\in R} c_k(i,j)$ and $M_R = \max_{(i,j)\in R} c_k(i,j)$. Let B_R be the interior boundary of R. Let S be the set of samples either: 1) inside R and within dS_i pixels of B_R or 2) outside R and within dS_o pixels of B_R. Finally, let G_R be the subset of S of "good" samples, i.e. $G_R = \{(i,j) \in S | c_k(i,j) < P_G \cdot m_R + (1 - P_G) \cdot M_R\}$.

Then R is post-processed if and only if $|G| \geq TG_k$ or $\frac{|G|}{|S|} \geq FG$. This determines which spots are bright and have to be inpainted, and as previously mentioned, the same idea can be used to detect extreme dark spots to inpaint.

5.6 Repairing of Extreme Regions by Image Inpainting

The very bright and dark areas of the image formed by bad edge localization have to be removed somehow so that a suitable reconstruction is obtained. The best approach is to regard these regions as damaged and try to reconstruct their appearance using surrounding image data. This is a problem that has only recently been put forward in the digital domain and a solution methodology called image inpainting proposed, e.g. [30]. The method in [30] relies on the propagation of isophotes, or level curves, and is based on a PDE formulation. Unfortunately, this inpainting technique leads to blurring of edges inside regions being inpainted, and since edge-blurring is exactly what we are trying to eliminate, this is not a suitable choice for an inpainting

algorithm.

Two other algorithms were however presented in [31] reconstruct edges sharply in the region being inpainted and so are potential candidates to be used in NEDMBA. These algorithms are based on the Mumford-Shah and Mumford-Shah-Euler models respectively. The Mumford-Shah-Euler model is a higher-order correction of the Mumford-Shah model which replaces the straight edge model in the Mumford-Shah model with smooth curves. Nevertheless, the increase in complexity caused by this correction makes the M-S-E model unwieldy, and because only relatively small regions in the image are being inpainted, the M-S model suffices.

Following the notation in [31], let Ω be the entire image and let D be the union of all the extreme bright and dark regions which are to be post-processed, or the inpainting domain. Suppose that u^0 is the known image function outside of D and that λ_D is the characteristic function of $\Omega \backslash D$ (equal to 1 on $\Omega \backslash D$, 0 otherwise). Let Γ be the edge set of the image and $H^1(\Gamma)$ be the Hausdorff measure of Γ, which is basically its length. Then, to inpaint an image with the Mumford-Shah model involves minimizing the following energy functional:

$$E_{MS}[u, \Gamma | u^0, D] = \frac{1}{2} \int_\Omega \lambda_D (u - u^0)^2 dx + \frac{\gamma}{2} \int_{\Omega \backslash \Gamma} |\nabla u|^2 dx + \alpha H^1(\Gamma). \tag{5.3}$$

To make this tractable, Γ is approximated with the edge signature function z which is 1 on the entire image except for a small ϵ neighborhood of Γ where it is near

0. Then using this edge signature function, estimates of a pair of the terms in the expression for E_{MS} can be made, namely:

$$\int_{\Omega\setminus\Gamma} |\nabla u|^2 dx \approx \int_{\Omega} z^2 |\nabla u|^2 dx \tag{5.4}$$

and

$$H^1(\Gamma) \approx \int_{\Omega} (\epsilon|\nabla z|^2 + \frac{(1-z)^2}{4\epsilon}) dx. \tag{5.5}$$

Therefore, the functional:

$$E^{\epsilon}_{MS}[u, z|u^0, D] = \frac{1}{2}\int_{\Omega} \lambda_D(x)(u-u^0)^2 dx + \frac{\gamma}{2}\int_{\Omega} z^2|\nabla u|^2 dx + \alpha\int_{\Omega}(\epsilon|\nabla z|^2 + \frac{(1-z)^2}{4\epsilon})dx, \tag{5.6}$$

must now be minimized.

Using the calculus of variations, the minimum is the solution of the following Euler-Lagrange system,

$$\lambda_D(x)(u-u^0) - \gamma\nabla\cdot(z^2\nabla u) = 0 \tag{5.7}$$

$$(\gamma|\nabla u|^2)z + \alpha(-2\epsilon\Delta z + \frac{z-1}{2\epsilon}) = 0. \tag{5.8}$$

This system is taken together with the following boundary conditions:

$$\frac{\partial u}{\partial \vec{n}} = 0, \frac{\partial z}{\partial \vec{n}} = 0, \tag{5.9}$$

where $\vec{n}$ is the outward unit normal from Ω.

This system can be solved iteratively using a sequential strategy. First, an initial guess for z is made based on the computed edge magnitudes from u, where u on D is uniformly randomly distributed and u outside of D is given by the original image. Then u is solved for from the first equation in the Euler-Lagrange system, and after this the value obtained for u is used to solve for z. This is done for as many iterations as needed for u and z to converge. In practice, 15 iterations were used in our experiments. When MATLAB was used to directly solve PDEs with the `assempde` command, and solutions found alternately for u and z, blurring of the entire image resulted. Therefore, instead a finite difference scheme is used, where for each of 15 loops, 5 iterations of solving for u with the given z are performed, and then 5 iterations of the solution of z are executed. It is thought that performing a similar procedure with `assempde` would provide the same quality of results, but this takes too much computation time, so the finite difference technique is preferred.

The solution of z, the edge signature function, is discussed first. From the second equation of the system, $M_u z = 1$ is obtained where $M_u = (1+\frac{2\epsilon\gamma}{\alpha})|\nabla u|^2 - 4\epsilon^2\Delta$. This equation must be discretized. Let $z_{i,j}^n$ be the value of z at pixel (i,j) and at iteration n. Then making the substitution $(\Delta z)_{i,j}^n = z_{i,j+1}^n + z_{i,j-1}^n + z_{i+1,j}^n + z_{i-1,j}^n - 4z_{i,j}^{n+1}$ yields:

$$(1 + 2\frac{\epsilon\gamma}{\alpha}|\nabla u|^2)z_{i,j}^{n+1} - 4\epsilon^2(z_{i,j+1}^n + z_{i,j-1}^n + z_{i+1,j}^n + z_{i-1,j}^n - 4z_{i,j}^{n+1}) = 1$$

$$\Rightarrow z_{i,j}^{n+1} = \frac{1 + 4\epsilon^2(z_{i,j+1}^n + z_{i,j-1}^n + z_{i+1,j}^n + z_{i-1,j}^n)}{1 + 16\epsilon^2 + \frac{2\epsilon\gamma}{\alpha}|\nabla u|^2}. \tag{5.10}$$

At each iteration, z and u are extended adiabatically, by padding with repetition for 1 pixel on each side to satisfy the boundary conditions.

To solve for u, the fact that the image outside of the inpainting region is already known is taken advantage of. For each iteration, u is kept unchanged outside of D. Inside D, the first equation of the Euler-Lagrange system becomes: $\nabla \cdot (z^2 \nabla u) = 0$. Away from edges, where z is close to 1, this equation becomes $\Delta u = 0$, or when discretized, $u_{i,j}^{n+1} = \frac{1}{4}(u_{i+1,j}^n + u_{i,j+1}^n + u_{i-1,j}^n + u_{i,j-1}^n)$. To allow z to exert control over the evolution of u, the right hand side of the previous equation is replaced with a weighted average as in [32] so that:

$$u_{i,j}^{n+1} = \frac{(z_{i+1,j}^n)^2 u_{i+1,j}^n + (z_{i,j+1}^n)^2 u_{i,j+1}^n + (z_{i-1,j}^n)^2 u_{i-1,j}^n + (z_{i,j-1}^n)^2 u_{i,j-1}^n}{(z_{i+1,j}^n)^2 + (z_{i,j+1}^n)^2 + (z_{i-1,j}^n)^2 + (z_{i,j-1}^n)^2} \tag{5.11}$$

is obtained. This weighted average is used to maintain edges inside D, without the image on one side of the edge propagating to the other side.

5.7 Choice of Parameters and Uninpainting

NEDI and inpainting are used only for levels 5 and above, where the first level is level 0. For levels 0 to 2, plain MBA is used, and for levels 3 and 4, blurring is implemented. It may seem counterintuitive to purposefully blur a reconstruction (or

at least an intermediate part of it), when the aim is to sharpen this reconstruction, but the explanation of this is done is quite simple. If the image is not blurred, then the sum of the coarser spline surfaces will approximate the sample intensities for those samples along an edge quite well (as in Figure 4.5), and so the edges in the final reconstruction will be represented by this sum of coarser spline surfaces, and thus be blurred. If a blur is performed, however, then the coarser surface will be more slowly changing and the sum of surfaces will still leave some residual at the samples close to the edge. Thus, when NEDI is used and when a zoom is performed, the overall edge which is represented by the sum of all the levels to that point will be stronger. The blurring is executed by first finding the function values at the control grid points by using Equation 4.5 with $s = t = 0$, and the control grid values as done in Section 5.3. Then instead of using NEDI to zoom on the image, the function for this level is found once again using Equation 4.5 except using the newly computed function values F_{ij} at the control grid points and not the control grid values, ϕ_{ij}. That is, the function for this level at all pixels in the image is found from the equation:

$$f(x,y) = \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)F_{(i+k)(j+l)}. \tag{5.12}$$

In the selection of regions to fill in, the following parameters were used (where for 256x256 images, there are 8 levels, 0 to 7): $DF_k = [40, 25, 18]$, $dS_i = 3$, $dS_o = 1$,

$P_G = 0.5$, $TG_k = [4, 2, 2]$, $FG = 0.4$. For inpainting, the parameters chosen were as follows: $\gamma = 0.0125$, $\epsilon = 1$ and $\alpha = 0.00025$. Finally, a good value for P_B was found to be 0.025. These settings were set experimentally and found to work with the entire test set of images.

Even with these parameters, however, sometimes a region is inpainted by mistake leading to aberrations in the reconstructed image. To stop this from happening, information from the original sample intensities in the region is used. Let there be N_R samples in the region R, let $\{o_i, 1 \leq i \leq N_R\}$ be the set of original sample values and $\{\pi_i, 1 \leq i \leq N_R\}$ be the intensity of the ith sample in the region after inpainting. Then if there is at least one sample in the region and median$(|o_i - \pi_i|) > U_k$, the region is uninpainted, which means all pixels are returned to their original c_k values. A good choice for the U_k's was found to be $[55, 35, 20]$.

5.8 Experimental Results

Figure 5.2 shows the reconstructions of regular MBA vs. NEDMBA for `peppers`, one of the 8 test images from Figure 3.7. The reconstructions start from 2523 skewness-based samples. The parameters for this sampling of `peppers` were the same as those used in Table 3.1: $\theta_s = [0.0025, 0.0063]$, $r = [15, 8, 2]$, $\theta_d = 6$, $\theta_v = 40$ and $f = 50$. The skewness-based samples were chosen to test NEDMBA since they result in the most blurring when reconstructed in MBA, and thus needed the most improvement.

Despite the fact that the edges from NEDMBA (Figure 5.2(d)) are visibly sharper than those from MBA (Figure 5.2(b)), the SNR of the NEDMBA result is lower (9.86) than that of MBA (10.89). This is because the intensities in the homogeneous regions are not as close to the original, but overall the NEDMBA image is superior, as more information is carried by the edges. The SNR is thus not a good measure of overall image quality, but it is included here as it is widely used. Figures 5.2(e) and 5.2(f) show zooms of the bottom right of the long vertical pepper at the left of the image in Figure 5.2(a). The lower edge in the zoom of the NEDMBA reconstruction Figure 5.2(f) is a bit jagged relative to Figure 5.2(e), but this lack of smoothness is not visible when the image is viewed at its normal size. On the other hand, the edges are visibly sharper for NEDMBA than for plain MBA. In Figure 5.2(c), the result of NEDMBA on `peppers` with the inpainting step omitted is shown. As can be seen, the use of inpainting is necessary in order to prevent the presence of bright spots, for example on the top-left corner of the large bottom-centre pepper.

Figure 5.3 shows the results of NEDMBA vs. the other algorithms. Here the adapted 4-NNI reconstruction of `femme` from 2108 skewness-based samples is also included. As with `peppers`, the MBA reconstruction from skewness-based samples in Figure 5.3(b) is very blurry. Adapted 4-NNI (Figure 5.3(c)) is an amelioration over this, but unfortunately the level curves on the face of the woman are very disturbing. NEDMBA (Figure 5.3(d)) is the best of all, with edges visibly sharper than those from MBA, superior details to MBA, for example the eyes and teeth, and without

any of the artifacts with adapted 4-NNI.

NEDMBA also works well with samples from Farthest Point Sampling or Faster Farthest Point Sampling. The outcome of tests on these samples is shown in Figure 5.4. The reconstruction in Figure 5.4(a) using MBA is very dark and blurred. The edges in Figure 5.4(b) are not smooth and very jagged, which means that FPS and Adapted 4-NNI are not compatible algorithms. The NEDMBA reconstruction of `femme` from FPS samples once again gives the best result - there are sharper edges and the eyes are brighter than in the MBA reconstruction, without the extreme jagged edges of the adapted 4-NNI reconstruction.

5.9 Conclusions

Experiments on a wide variety of images show that NEDMBA works on many different types of images, for example natural and medical ones, and can be used for improved image compression; that is using the same amount of data as MBA, it is possible to yield improved reconstructions from irregular samples. It is still somewhat slow (on the order of minutes, not seconds for one image), so it remains an open problem to try to improve its efficiency. Instead of NEDI, a different 2x interpolator could be used. As well, the use of weighted B-splines [29] is an attractive option. Finally, using a gridding method other than MBA followed by function evaluation at the control grid points is another possibility which could be explored.

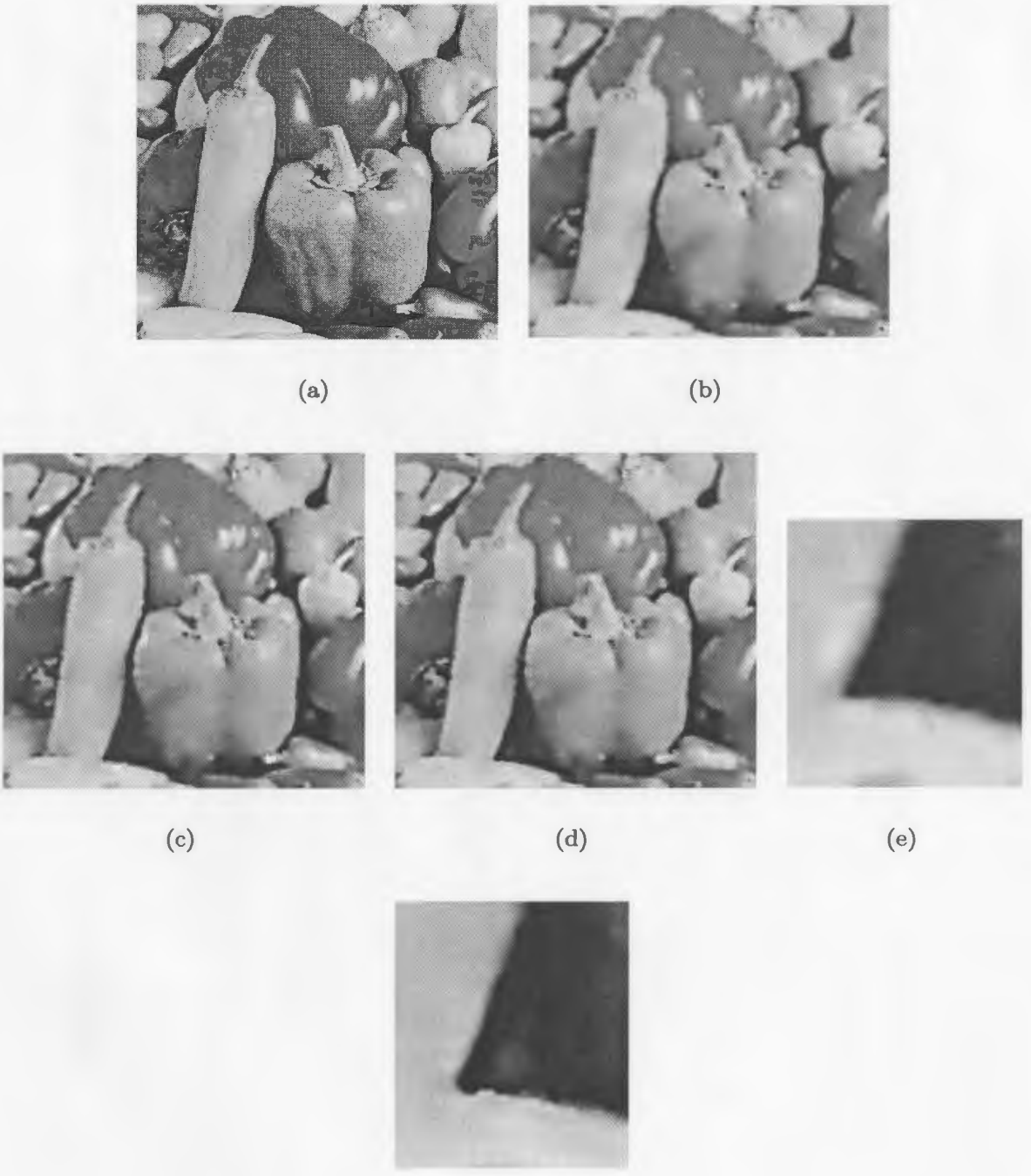
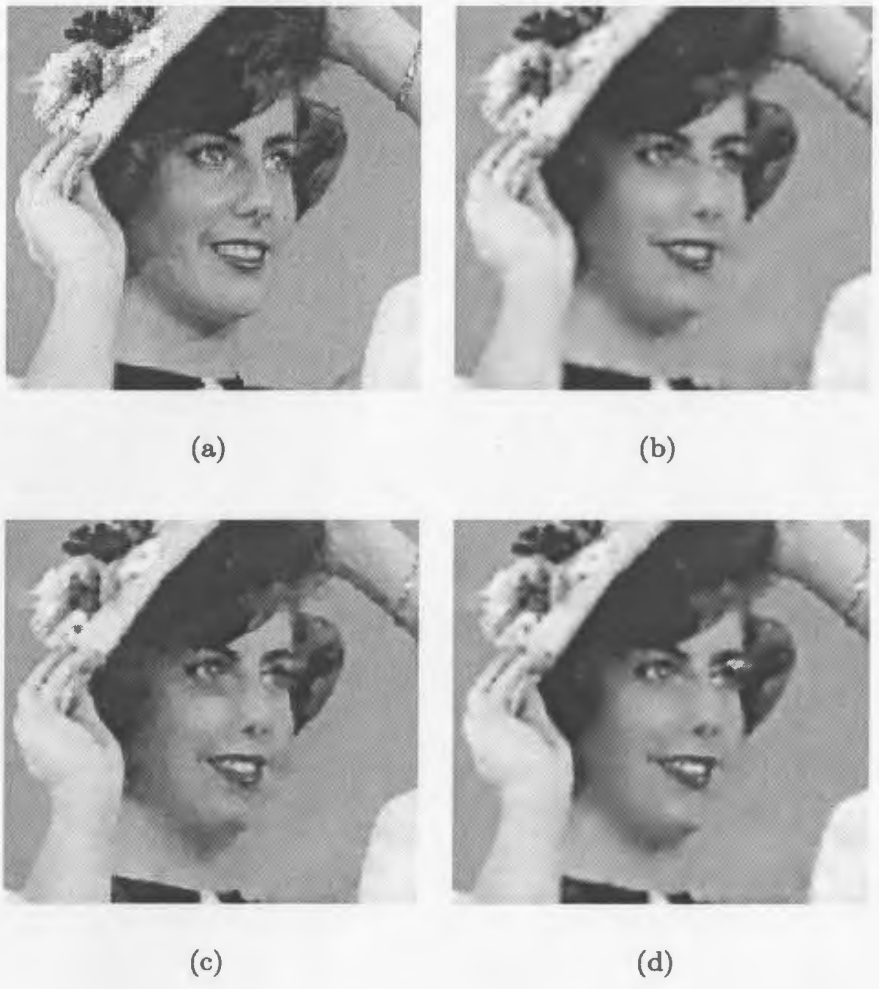


Figure 5.2: (a) Original **peppers** image, (b) Entire MBA reconstruction from skewness-based samples, (c) NEDMBA reconstruction without inpainting, (d) NEDMBA reconstruction, (e) Zoomed portion of MBA reconstruction from (b), (f) Zoomed portion of NEDMBA reconstruction from (d).



(a) (b)

(c) (d)

Figure 5.3: (a) Original `femme` image, (b) MBA reconstruction of `femme` from skewness-based samples, (c) Adapted 4-Nearest Neighbor Interpolation of `femme` from skewness-based samples, (d) NEDMBA reconstruction from these samples.

(a) (b)

(c)

Figure 5.4: (a) MBA reconstruction of `femme` from Farthest Point samples, (b) Adapted 4-Nearest Neighbor Interpolation of `femme` from Farthest Point samples, (c) NEDMBA reconstruction from these samples

Chapter 6

Farthest Point Halftoning

6.1 Introduction

As mentioned in the introduction, digital halftoning refers to the display or printing of a continuous-tone or many-level image to fewer levels, often only two. It was found by Ulichney that the most pleasing halftoned (also known as dithered) images were the ones with dots arranged in a high frequency, or blue noise, pattern without low frequency artifacts [22]. Irregular sampling can be applied to this problem, so this thesis is rounded out by doing this.

In this thesis, the halftoning of gray-scale images and not color images is exclusively dealt with. As well, only halftoning algorithms which are point processes are looked at. A point process is used when each pixel in the image to be halftoned is thresholded against a fixed array of the same size as the image, and is taken to be

white if it is greater than the corresponding element of the array, or black otherwise. The advantage of such algorithms is their speed; there is the one-time cost of generating the threshold array, but after this, the point-by-point comparison can be done very rapidly for all images in graphics hardware.

For each gray level, there is what is called a dot profile of the threshold array, which is formed by using the threshold array to halftone an array with all elements at that given constant gray level. The dot profile can be considered to be a binary image, with black pixels equal to zero, and white pixels equal to 1. Then, the gray level is the average of these 1's and 0's over the entire array, where a gray-level image with gray levels in the range $[0, G-1]$ is normalized to range [0,1]. Because of the definition of the dot profiles, if a pixel is 1 for one gray level g_1, it has to be 1 for all gray levels g_2 with $g_2 > g_1$.

For point processes, in order to have good quality halftoned images, the binary dot profiles should have blue noise spectra. In Figure 6.1, the spectrum for the initial binary pattern from the Void and Cluster method (the dot profile for gray level 0.5) is shown. In fact, this is a radially averaged power spectrum, which provides us with most of the information since the 2-D spectrum is mainly isotropic, with a low amount of directional dependence. A dot profile with such a spectrum has fewer noticeable low frequency structures which take away from halftone quality. A radially averaged power spectrum is obtained first by taking annuli of all radii that lie in the image, and averaging the power of all the pixels that are in an annulus of a given

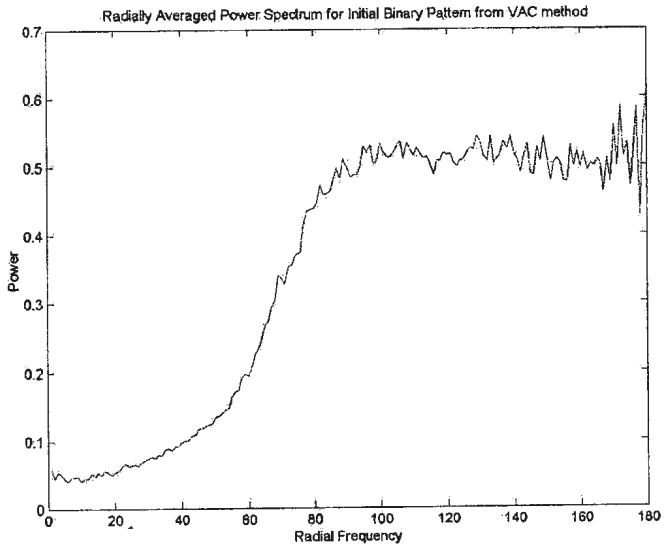


Figure 6.1: A typical blue noise spectrum

radius away from the centre of the spectrum (the DC component). A pixel is taken to be in an annulus of a specific integer radius if the floor function of its radius from the centre of the image is that integer.

In the next section, the details of three existing algorithms, the Modified Blue Noise Mask, the Void and Cluster method and Linear Pixel Shuffling halftoning are given, after which a new technique called Farthest Point Halftoning is put forward based on the Faster Farthest Point Sampling (FFPS) algorithm of Chapter 3, which gives better results compared to the three above-named methods.

6.2 Existing Algorithms

6.2.1 The Modified Blue Noise Mask

The Blue Noise Mask (BNM) [33] was the first attempt to simulate the performance of neighborhood processes like error diffusion (which are of very high quality) with point processes (which are much quicker to implement, at least once the threshold array has been formed). The early BNM algorithm attempted to force the dot profiles to have blue noise characteristics (by approximating a given gray-level specific blue noise spectrum) using a filtering and swapping approach. Three years later, an improvement to this algorithm was reported [34] called the Modified Blue Noise Mask (MBNM) which was simpler conceptually, and gave better halftoning results. The latter algorithm is briefly described here.

The Modified Blue Noise Mask starts off with a blue noise pattern for an intermediate gray level g_i, $0 \leq g_i \leq 1$. Then dot profiles are generated for all gray levels above g_i, and subsequently for all those below. The final threshold array can be considered to be the sum of all these binary dot profiles. The initial blue noise pattern is generated using an iterative process called the BIPPSMA (BInary Pattern Power Spectrum Manipulating Algorithm) starting from a white noise arrangement of 1's and 0's. The centres of the largest clumps of 1s and 0s in the binary pattern are found by locating the extrema of a low-pass filtered version of the binary pattern at that iteration, after which these centres are swapped. This low-pass filtering is done

in the frequency domain via FFTs. The M 1s in the binary pattern with the highest filtered values and the M 0s with the lowest filtered values are swapped, where the value of M starts off from an arbitrary value, say the highest power of 2 less than the number of minority pixels in the binary pattern. The mean squared error (MSE) is calculated for the filtered pattern with respect to the given gray level; if the MSE has gone up, then the value of M is halved, and otherwise M is not changed.

The low-pass filter used takes the principal frequency for the gray level g as a parameter, the principal frequency being given by the formula $f_g = \min(\sqrt{g}, \sqrt{1-g})$. Two different shapes of low-pass filters were suggested in [34], one is a Gaussian, and the other a Butterworth filter. The Gaussian is simpler and yields better dot profiles, so this is chosen for our discussion. The form of the Gaussian function in the frequency domain is

$$F(u,v) = e^{-\frac{u^2+v^2}{2\sigma^2}}, \tag{6.1}$$

where u and v are frequency coordinates, and $\sigma = 0.4 \times f_g$. In our tests, an anisotropic version of this filter, as suggested in [34]:

$$F'(u,v) = F(u,v)[1 + 0.2\cos(4\theta)] \tag{6.2}$$

was used. Here, θ is the angle of the frequency position with respect to the the central DC point, and using $F'(u,v)$ increases the resulting energy in the mask in diagonal directions, to which the human eye is less sensitive, improving halftone quality.

The upwards and downwards construction of the dot profiles is basically the same as BIPPSMA, except that in the swapping stage, it is ensured that the stacking constraint is always satisfied. Also, for the downwards progression, the binary pattern is inverted so that the Gaussian filter with the same principal gray level-dependent standard deviation σ can be used [1].

6.2.2 The Void and Cluster Method

The Void and Cluster (VAC) method for halftoning was first put forward by Robert Ulichney in 1993 [36]. It tries to eliminate unwanted clumps and empty regions (i.e. without 1s) in the halftone threshold array and thus in the halftoned image itself. Unlike the Modified Blue Noise Mask (MBNM), it does this by filtering in the spatial domain instead of the frequency domain. Its description is much simpler than that of the MBNM, and it is also faster. Like the MBNM, the VAC algorithm needs to start with an initial binary pattern at an intermediate gray level g_i. This is done via the Initial Binary Pattern Generator. Notice once again that each gray level is labelled by the fraction of pixels contained in it that are 1s. The Initial Binary Pattern Generator starts off with an arbitrary pattern, say a white noise pattern with a fraction g_i of pixels turned on. Then clusters (groups of 1s or "on" pixels) are broken up iteratively and the presence of voids (areas without any "on" pixels) is reduced. Clusters and voids are found by computing a circular convolution of

[1]The lack of inversion of the binary pattern on the downwards progression was why it was incorrectly reported in [35] that the parameters in [34] did not work.

the binary pattern $b(x, y)$ with a Gaussian filter for every position in the array. An explanation of why a circular convolution is taken is given below. This can be done very rapidly using a look-up table for the Gaussian function, and using the fact that the entries of the look-up table are only ever added together, and not ever multiplied by any numbers other than 1 or 0. The Gaussian filter is of the form $f(x, y) = e^{\frac{-r^2}{2\sigma^2}}$, where $r^2 = x^2 + y^2$. A good value of σ was found to be 1.5 by Ulichney [36]. The filter can also be restricted to have a finite support (the region where it is non-zero), since its elements are very close to 0 for larger r. At each stage, this convolution, also known as a cost function, is calculated for all pixels in the image (in fact the sum is not recomputed for all pixels, just appropriately incremented or decremented for surrounding pixels of the position where a change was made in the array depending on whether respectively a pixel was either turned on or off). The convolution is:

$$C(x, y) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} b((M + x - m) \bmod M, (N + y - n) \bmod N) f(m, n). \quad (6.3)$$

A circular convolution is used in order to allow smaller halftone arrays to be generated which can be tiled over a larger image, so that no boundary effects are created. For a given (x, y) it is clear that the more pixels that are on, the higher C becomes. Also, the closer an "on" pixel is to (x, y), the more that C will increase. This is because the maximum of f occurs at $(0, 0)$, and this is multiplied by $b(x, y)$, so $b(x, y)$ has the most influence on $C(x, y)$. Since f tapers off, it is easy to see

that in general $b(\hat{x}, \hat{y})$ will have more of an effect as $(\hat{x}, \hat{y})$ approaches (x, y). So the minimum of C can be viewed as being the pixel that is farthest away from all "on" pixels, or the centre of the largest void, while the maximum of C can be regarded as being the centre of the tightest cluster. In the Initial Binary Pattern Generator, the centre of the tightest cluster is changed from 1 to 0, thereby somewhat breaking the cluster up, and then the centre of the largest void is found and is changed to a 1. In effect, the centres of the largest voids and clusters are being swapped. This process ends when after changing the centre of the tightest cluster to a 0, the centre of the largest void corresponds exactly to the pixel that was just modified. This means that there was no change in this iteration, and so the process has converged.

The rest of the Void and Cluster algorithm is quite straightforward. First, the dot profiles for all gray levels greater than g_i are built, and then they are constructed for those levels less than g_i. Figures 6.2 and 6.3 (adapted from [37]) show the upward and downward progression processes respectively.

6.2.3 Using a Linear Pixel Shuffling Screen

Linear Pixel Shuffling (LPS) [19] was already discussed in Subsection 2.3.2, in which skewness-based sampling was made progressive. LPS can also be used to create a halftone screen for a dispersed-dot ordered dither algorithm. The algorithm is the same as the upwards procession in Figure 6.2, except that it starts at level $g_i = 0$, and the initial binary pattern is all zeros. The flow chart is followed exactly, other

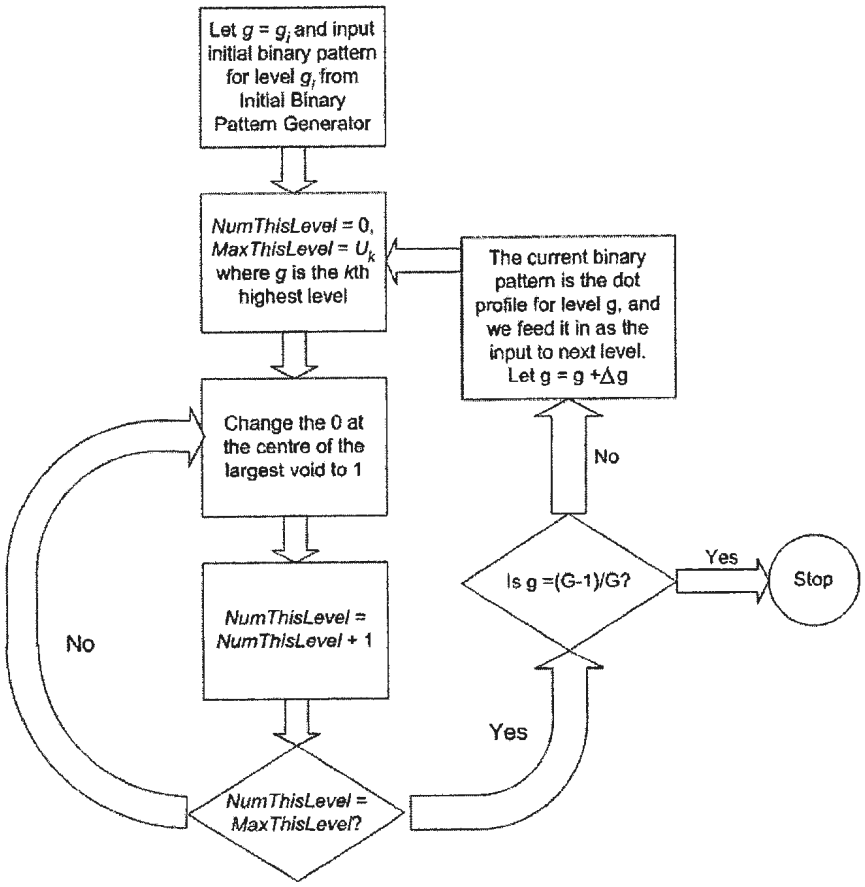


Figure 6.2: Construction of Void and Cluster method dot profiles for levels above g_i ($\Delta g = \frac{1}{G}$).

than the fact that instead of placing a 1 in the largest void, we change zeros to ones in the LPS order, using the matrix $M = \begin{pmatrix} G_{-n+1} & G_{n-3} \\ G_{-n} & G_{n-2} \end{pmatrix}$, as given in Chapter 2.

6.2.4 Kang's Microcluster Halftoning

Kang outlines an algorithm in [13] for creating dispersed-dot ordered-dither arrays of arbitrary dimensions. Kang's algorithm is used for microcluster halftoning, a cross

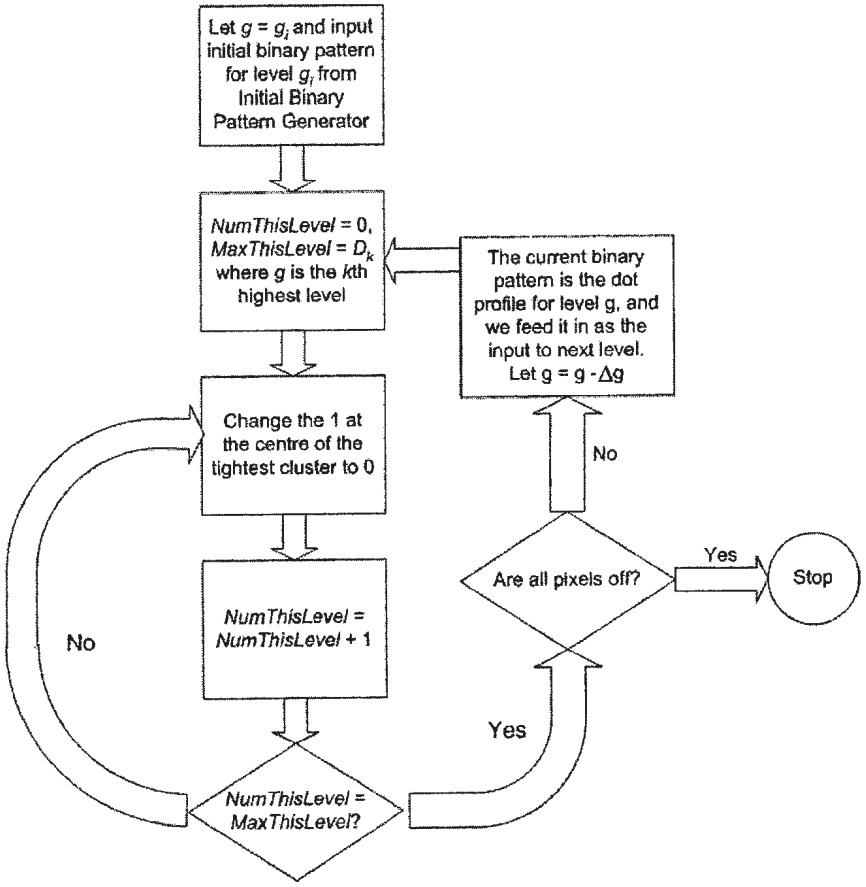


Figure 6.3: Construction of dot profiles for levels below g_i ($\Delta g = \frac{1}{G}$).

between dispersed-dot and clustered-dot ordered dither. In his application, only very small (e.g. 5x5) masks need to be formed.

Thus, efficiency was not a concern for Kang. His algorithm generates the dot profiles of the threshold array in an upwards fashion, starting at level 0 with an all zero mask; at each stage it chooses the pixel which is dispersed the most with respect to all the pixels previously turned "on" (or to a 1). Whereas in Linear Pixel Shuffling (LPS) pixels can be visited in the order of their indices in a table, Kang's algorithm

chooses the next pixel to be the one with the smallest calculated *dispersion*. This dispersion is a function of the distances to the four closest pixels which are already 1's; this would be computationally inefficient for larger masks due to the fact that the four nearest neighbors would have to be recalculated for all pixels after a new pixel is turned on. Our approach, based on a generalization of Faster Farthest Point Sampling (FFPS) can solve this problem very quickly, as we discuss in Section 6.3.

Kang defined the dispersion of a pixel to be

$$\Lambda(i,j) = \sum_{k=1}^{4} \frac{|d_k(i,j) - \bar{d}(i,j)|}{\bar{d}(i,j)}, \tag{6.4}$$

where $\bar{d}(i,j)$ is the average distance to the four nearest neighbors. This tends to be low for positions which are far away from "on" pixels and where the variance of the distances to pixels already turned on is small. Unfortunately, the dispersion measure does not distinguish between pixels which are equidistant from their four nearest neighbors, since regardless of what this distance is, the dispersion is 0. For example, if the four nearest neighbors of two different pixels are at distances of (1,1,1,1) and (4,4,4,4) respectively, they are treated identically.

Kang's algorithm sequentially selects pixels with the lowest dispersion until there are only single pixel "holes", whose 4 immediately adjacent horizontal and vertical positions are on. Then, the holes which are closest to the others and with the smallest dispersion are selected. The problem with this approach is that even before all the

remaining pixels to be filled are "holes", there are many ties, as many candidate pixels have the same 4 closest distances. In effect, for the higher gray levels, the dispersion contains less information about the best pixel to choose next.

Next our new halftoning technique called Farthest Point Halftoning is discussed which is based on Kang's microcluster halftoning but solves many of the problems with his method.

6.3 Farthest Point Halftoning

Farthest Point Halftoning is a new halftoning algorithm based on Kang's [13] microcluster halftoning. The Farthest Point Sampling method was introduced in [14] for effective irregular sampling of an image and was described in full detail in Chapter 2 of this thesis. To exploit it for halftoning, the key observation is that in general a good set of irregular samples will have a blue noise spectrum, which is the desired characteristic of the spectra of good dot profiles for halftoning. An irregular sampling method applied to halftoning might start from the dot profile for gray level 0 and work its way up, at each point choosing the next pixel to be the one that is farthest from all previously selected pixels. This idea is not practical however, because of the extreme amount of time needed to sample the entire image for the formation of all the dot profiles of all gray levels. Also, not enough information is held by the distance to the closest pixel already selected (as with the strictly upward

Kang algorithm), especially for higher gray levels where the closest distance to an "on" pixel is 1 for a large number of the points not chosen to that stage.

In the next subsection, our new Farthest Point Halftoning (FPH) algorithm is introduced, and its performance is compared to those of existing algorithms: the Modified Blue Noise Mask (MBNM) [34], the Void and Cluster (VAC) method [36], and halftoning with an LPS threshold array [19].

6.3.1 The Algorithm

For simplicity, it is assumed that the dot profiles are generated upwards from an all-zero (except for four randomly chosen "on" pixels) level to an intermediate level g_i. This will be made more clear below, but the important thing to note is that it is being assumed in this short discussion that pixels are being turned from "off" to "on". To begin, $d_k(i,j)$ is defined as as the distance (Euclidean) of the kth closest "on" pixel to (i,j). Kang's dispersion is a function of $\{d_i\}_{i=1}^4$. It may seem that finding the four closest distances for any pixel would be inefficient, but this is not the case. A generalization of the Faster Farthest Point Sampling algorithm can be used, with the small change that we appropriately update the four closest distances for all pixels in an image I which are within $\max_{(i,j)\in I} d_4(i,j)$ of the pixel just changed to 1. This makes it possible to generate reasonably sized dither arrays, for example 128x128, in about the same time as or faster than the existing MBNM and VAC algorithms. Though this is a good characteristic, it is not vital, since the threshold

array is only formed once, after which it can be reused for multiple images. However, if it is wished to halftone images of different gray level depths, then this becomes more important. In addition, our threshold array of a fixed size can easily be tiled for larger images using a toroidal topology.

Because of the problems with Kang's dispersion measure, a new dispersion measure Λ' is developed:

$$\Lambda'(i,j) = w_1\Lambda(i,j) + w_2 \sum_{k=1}^{4} e^{\frac{-d_k^2(i,j)}{2\sigma^2}} + \frac{w_3}{d_4(i,j)} + w_4 cb(i,j) + w_5 o(i,j) + w_6 \Delta(i,j), \quad (6.5)$$

where Λ, cb, o and Δ are defined below.

The weights w_i used in the above equation should be selected as a function of the size of the dot mask; for example, in all our experiments with 128x128 grids a convenient choice was $w = [0.8, 1.6, 1.0, 0.45, 0.7, 0.6]$. These weights were experimentally determined to give good results, along with a value of σ of $\sqrt{3}$. The modified dispersion in [38] was of the same form as Equation 6.5, but included none of the cb, o or Δ terms. In Equation 6.5,

$$cb(i,j) = \begin{cases} 1 & \text{if turning } (i,j) \text{ on (or off) forms a checkerboard pattern} \\ 0 & \text{otherwise} \end{cases} \quad (6.6)$$

$$o(i,j) = \begin{cases} 1 & \text{if } d_1(i,j) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

$$\Delta(i,j) = \begin{cases} 1 & \text{if } d_1(i,j) = \sqrt{2} \\ 0 & \text{otherwise} \end{cases} \tag{6.8}$$

The $\frac{1}{d_4(i,j)}$ term is included in order to reduce the appearance of checkerboard patterns, for example when switching a pixel on or off with all closest neighbors $\sqrt{2}$ away. However this does not prevent the formation of checkerboards by turning on or off pixels in other parts (not the centre) of the texture. So the checkerboard suppression term $cb(i,j)$ is added. That is, $cb(i,j) = 1$ if turning (i,j) on forms a checkerboard pattern in the upwards progression, or if turning (i,j) off forms a checkerboard pattern in the downwards progression. If this checkerboard suppression term is not included, then the dispersion of the pixels forming the checkerboard term is too low. Thus, the formation of checkerboard patterns not brought into existence by turning on the middle pixel is too favorable at levels far away from the middle level, where these patterns are better tolerated. Figure 6.4 shows an example of a pixel whose cb value is 1. When turning on or off a pixel it also has to be checked whether some pixels which before had $cb(i,j) = 1$ have now changed to have $cb(i,j) = 0$ and vice versa.

But if checkerboard patterns are penalized, then horizontal and vertical arrangements also become too highly favoured. So another penalty term for the formation

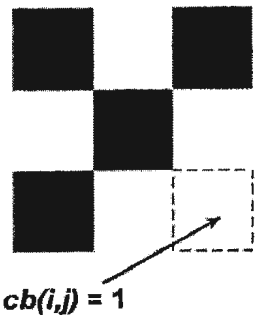


Figure 6.4: Example of a pixel forming a checkerboard

of these arrangements is used, which can be easily identified by checking to see if the closest distance to a pixel with the same binary value (as the value that we wish to change the current pixel under consideration to) is 1. The same is done for diagonal configurations by penalizing arrangements with closest distance equal to $\sqrt{2}$. Observe that the exponential terms used in the computation of Λ' are similar to the Gaussian filter in the VAC method [22], except that a larger standard deviation is used. We have optimized the parameters for a 128x128 mask size, but have not considered how the parameters depend on the dimensions of a general-sized array.

Additionally, because of the problems with a strictly upward progression in Kang's method, a two-step procedure is used around an intermediate level g_i in the $[0, G-1]$ luminance range of the original image to create the dot profiles. If $g_i = \lfloor \frac{G}{2} \rfloor$, as is recommended, then dispersions are always taken with respect to minority pixels. The two-step process is now described in more detail:

1. Create the dot profiles for all levels up to and including an intermediate level g_i, starting from level 0, picking the pixel with the lowest dispersion at each

stage. Start off with 4 randomly pixels turned on.

2. Build the dot profiles from level $G-1$ down to level g_i+1. Note that the dot profiles must satisfy a stacking constraint. So whenever a pixel is turned off, it must be already off in the dot profile for level g_i. The downward process starts with an initial pattern for level $G-1$ with all pixels on except for four random pixels chosen from those which are off at level g_i. Then pixels are turned off which have the lowest dispersions (Λ') with respect to off pixels. When enough pixels have been turned off in a level ($\sim \frac{mn}{G}$ for an $m \times n$ image), the level is decremented, and so on until the dot profile for level g_i+1 is formed.

Finally all the individual dot profiles are summed to produce the threshold array against which an input gray-level image is compared for halftoning. In fact, by the definition of the dot profiles of the threshold array, the dot profiles should be added to form the sum array $s(i,j)$, and then inverted to get the threshold array using the formula $t(i,j) = M - s(i,j)$, where M is the maximum gray level. However, inverting does not change the power spectrum, so we can just use $s(i,j)$ as our final mask.

The name Farthest Point Halftoning (FPH) is given to the entire process, that is the threshold array generation followed by the actual halftoning. Of course, as already mentioned, the threshold array, once generated, can be used for numerous halftonings; it may also be used in a tiled implementation against a larger image, as

in the results in the figures of the next section.

6.4 Results

6.4.1 Frequency Weighted Mean-Squared Error (FWMSE)

The most widely used metric for halftone quality is Frequency Weighted Mean-Squared Error (FWMSE), which is described in [13, 39]. Before proceeding, the caveat must be made that the FWMSE is not a particularly effective measure of halftone quality, partly because it is a global measure. However, this measure is used in Section 6.4.2, despite its limitations. This quantitative comparison must be tempered by visual inspection of the real halftone results, in order to judge the effectiveness of each of the halftoning methods.

The FWMSE metric in the form we use (see Equation 6.9) measures the difference between an original contone image and its respective halftone, according to a model of perception by the Human Visual System (HVS). More precisely, if o is the original $M \times N$ contone image, and h its halftone, then the FWMSE $E(o,h)$ is given by the equation

$$E(o,h) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}[(v * (o-h))(m,n)]^2 = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}|V(k,l)\cdot(O(k,l)-H(k,l))|^2 \tag{6.9}$$

The function v is an impulse response approximating the behaviour of the Human

Visual System (HVS) by an LSI system response, and the functions denoted by capital letters are the Discrete Fourier Transforms of the corresponding lowercase functions. The time and frequency domains in Equation 6.9 are linked by Parseval's Theorem. The frequency response of the HVS uses the modified Mannos-Sakrison visual model [13, 39], which is given by:

$$A(f_r) = \begin{cases} 1 - 0.00242 f_r & \text{if } 0 \leq f_r < 7.891 \\ \tilde{A}(f_r) & \text{if } f_r \geq 7.891 \end{cases} \tag{6.10}$$

$\tilde{A}(f_r)$ is the Mannos-Sakrison visual model on which the modified Mannos-Sakrison visual model is based. This function, found by regression, is given by:

$$\tilde{A}(f_r) = 2.6(0.0192 + 0.114 f_r) e^{-(0.114 f_r)^{1.1}}. \tag{6.11}$$

Then $V(k,l) = A(\sqrt{f_k^2 + f_l^2}), 0 \leq k \leq \lfloor \frac{M}{2} \rfloor, 0 \leq l \leq \lfloor \frac{N}{2} \rfloor$, with the usual DFT symmetry conditions: $V(M-k,l) = V(k,N-l) = V(M-k,N-l) = V(k,l)$ for these values of k and l. $f_k = 2DR\tan(0.5°)k/M$ and $f_l = 2DR\tan(0.5°)l/N$, where D is the viewing distance in inches and R is the printing or display resolution in dots per inch [39, 40]. For our calculations the values $R = 300$ dots/in and $D = 12$ in are used. A plot of the Human Visual Frequency Response is given in Figure 6.5. Note that it is lower in the diagonal orientations than the horizontal and vertical ones. Now that the FWMSE metric has been fully defined, it is applied to the

quantitative evaluation of the quality of halftone masks, as well as the comparison of the qualitative aspects of halftones generated with these masks.

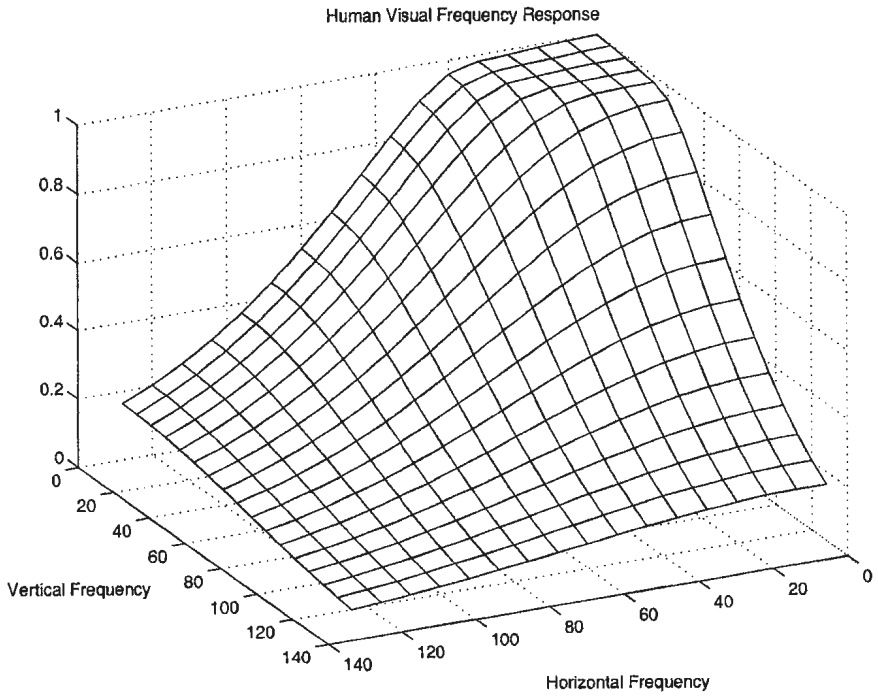


Figure 6.5: The Modified Mannos-Sakrison Model of the Human Visual Frequency Response

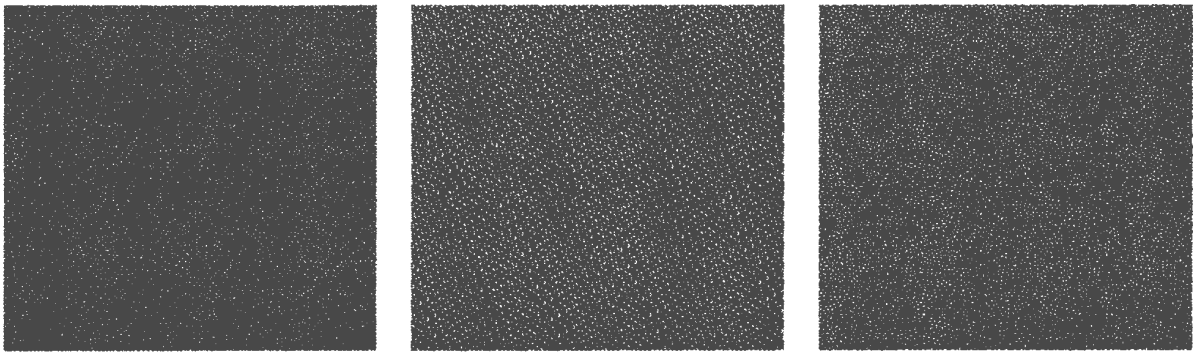
6.4.2 Halftoned Images and Quality Comparison

Figure 6.6(a) is a 256-gray level Modified Blue Noise Mask tiled to 256x256 pixels from a 128x128 mask. Figure 6.6(b) shows a 256x256 mask formed using Linear Pixel Shuffling Halftoning. In Figure 6.6(c), a 128x128 256-gray level mask generated by the Void and Cluster algorithm tiled to 256x256 pixels is shown. Finally, another 256x256 256-gray level mask formed by tiling a smaller 128x128 mask from Farthest

Point Halftoning is shown in Figure 6.6(d).

To compare the quality of these masks, the FWMSEs of the dot profiles for every 8 levels are looked at, starting from level 8 and ending at level 248. This gives us a good idea as to how the different halftoning algorithms perform for the entire range of possible gray levels, which may not be possible when looking at a given specific image. These FWMSE calculations are done on the 128x128 masks, not the tiled versions. The FWMSEs of the LPS halftoning mask are also computed, even though the entire array is not meant to be tiled, and the periodic nature of the DFT means that this tiling is assumed. This may affect the FWMSE results for this mask, though the situation should not be too grave, because the FWMSE is a global measure, and there are only problems on the boundary of the mask. Figure 6.7 gives a plot of the FWMSEs for the range of gray levels.

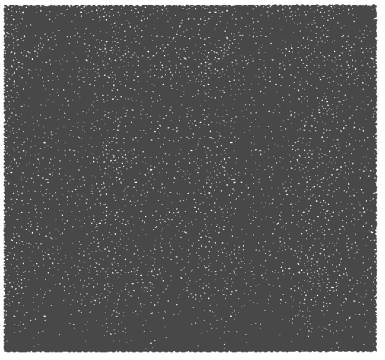
From Figure 6.7, it is observed that the lowest and thus the best FWMSE curve is the one from the Modified Blue Noise Mask, followed closely by the LPS mask, then that of our new algorithm, Farthest Point Halftoning, and finally the Void and Cluster curve. However, it is well known [39] that the FWMSE is not a completely accurate quality measure due to its averaging effect. Thus, all our test images are halftoned and some representative results (those from `cameraman` and `femme`, Figures 6.8 and 6.9 respectively) are shown, from which it is clear that FPH gives results which are superior. Measures that better reflect the characteristics of the halftone other than the FWMSE can be used, e.g. the texture metric of [41], but this example



(a) A typical 256x256 256-gray level Modified Blue Noise Mask, formed by tiling a smaller 128x128 array

(b) A typical 256x256 256-gray level threshold array created using Linear Pixel Shuffling

(c) A typical 256x256 256-gray level Void and Cluster threshold array, formed by tiling a smaller 128x128 array



(d) A typical 256x256 256-gray level Farthest Point Halftoning threshold array, formed by tiling a smaller 128x128 array

Figure 6.6: Threshold arrays from different halftoning algorithms (printed at 1200 dpi)

has the limitation of only being able to identify the presence of specific textures which have to be given, so this concept was not pursued in this thesis.

In Figure 6.8, showing halftone results of `cameraman`, it can be observed that

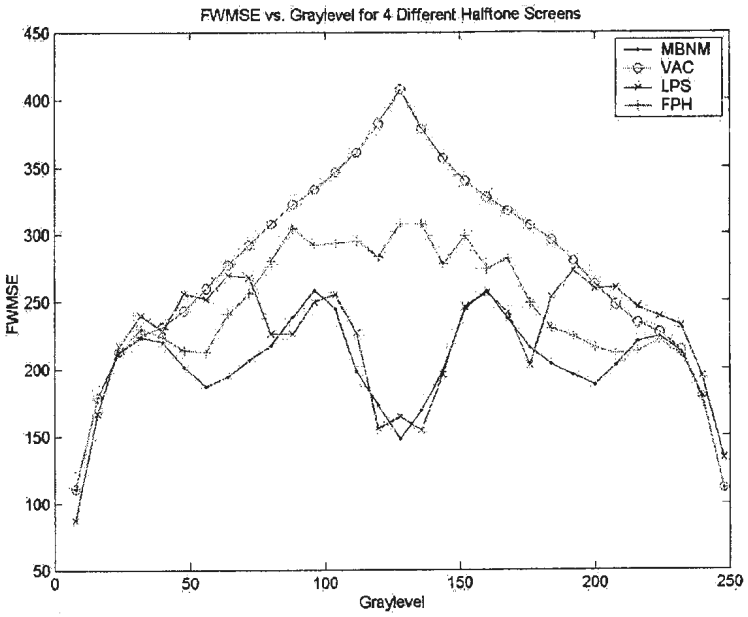


Figure 6.7: FWMSEs of dot profiles of different masks

the halftoned version using the Modified Blue Noise Mask (Figure 6.8(b)) is of quite poor quality. The transition from the slightly brighter part of the sky above the cameraman's head to the section of the sky in the right of the image is not very smooth, and the grass is not as uniform as it is in the original contone image. The halftone results for `femme` with the MBNM (Figure 6.9(b)) are equally bad - the face is splotchy and there are snake patterns (long connected white patterns) in the background. The Linear Pixel Shuffling halftones for both `cameraman` (Figure 6.8(c)) and `femme` (Figure 6.9(c)) contain disturbing and disruptive regular textures and patterns, which are especially evident in the uniform regions of the images. The Void and Cluster algorithm gives the best results out of the three existing halftoning

(a) Original 256 gray-level `cameraman` image

(b) Halftoned `cameraman` using the Modified Blue Noise Mask

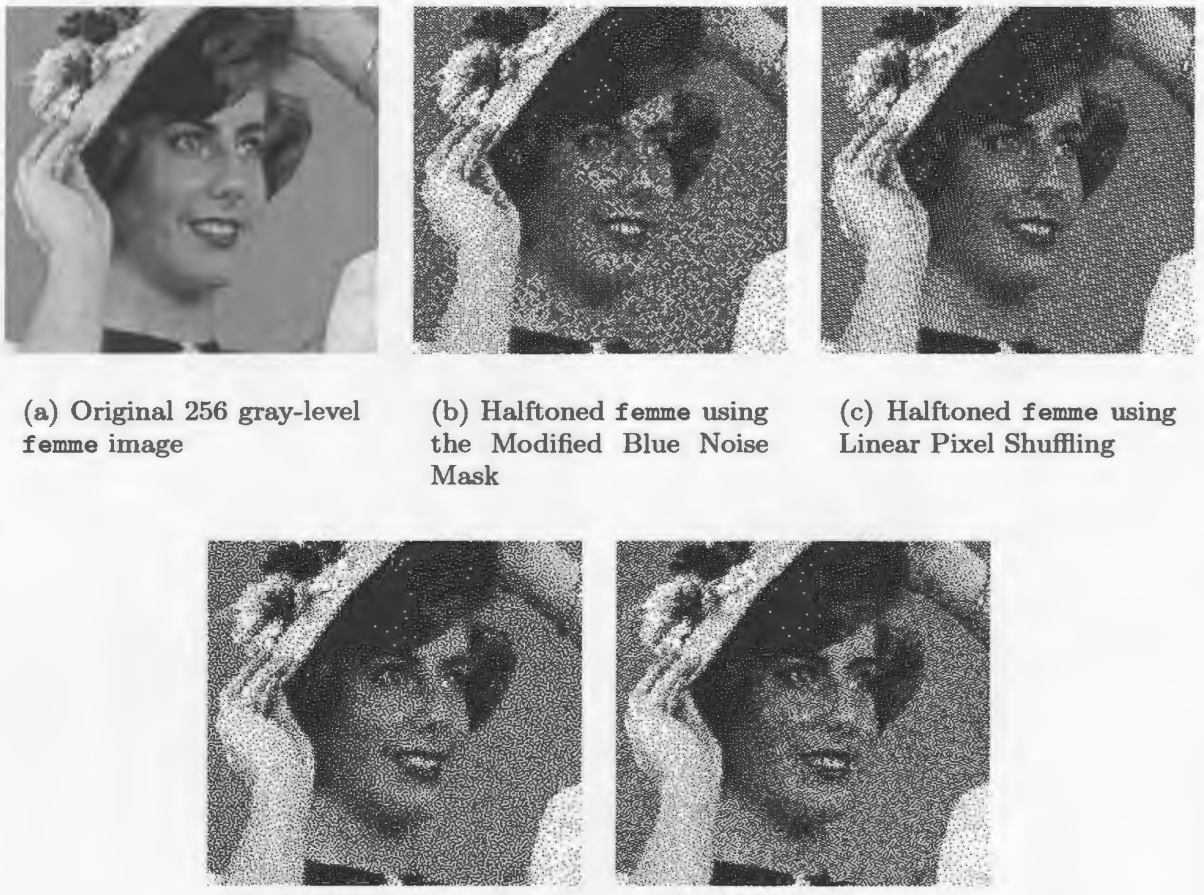
(c) Halftoned `cameraman` using Linear Pixel Shuffling

(d) Halftoned `cameraman` using a Void and Cluster threshold array

(e) Halftoned `cameraman` using Farthest Point Halftoning

Figure 6.8: `cameraman` image halftoned using screens from different algorithms (printed at 1200 dpi)

methods, but there is still evidence of some textures. For example, there are some coral patterns on the woman's face in `femme` (Figure 6.9(d)) and for `cameraman` (Figure 6.8(d)), these textures appear in the same problem areas as for the MBNM. For these two images, FPH (Figures 6.8(e) and 6.9(e)) gives the best results as none of the obvious patterns apparent for the other three algorithms appear in the halftone.



(a) Original 256 gray-level `femme` image

(b) Halftoned `femme` using the Modified Blue Noise Mask

(c) Halftoned `femme` using Linear Pixel Shuffling

(d) Halftoned `femme` using a Void and Cluster threshold array

(e) Halftoned `femme` using Farthest Point Halftoning

Figure 6.9: `femme` image halftoned using screens from different algorithms (printed at 1200 dpi)

Similar results are obtained for other images in the test set.

6.5 Conclusions

Farthest Point Halftoning was introduced in this chapter and found to perform better than the existing standards. Apparent artificial structures and textures are not introduced in the halftones. One possible improvement is the use of Manhattan distance instead of Euclidean distance - this would favor diagonal patterns to which the HVS is less sensitive. However, even without this potential improvement, the quality of halftoned images is very high. Better dispersion measures may also be possible, and it could be advantageous to have a gray level-dependent dispersion function. It is also important to derive parameters for a mask of general size, though this was not done here.

Chapter 7

Conclusions

7.1 Discussion of Results and Observations

This thesis has developed new techniques for the irregular sampling of digital images, the reconstruction of images from these irregular samples, and the bilevel halftoning of images based on irregular sampling. These new techniques were compared to existing methods and gave favorable results.

In Chapter 2, the Farthest Point Strategy and skewness-based sampling for irregular sampling of digital images were presented. In experiments in Chapter 4, FPS was shown to perform poorly on graphical images, especially when used with adapted 4-Nearest Neighbor Interpolation. Skewness-based sampling had the problems of the need of image-specific parameters and lack of flexibility in choosing the bitrate. In order to address these problems, a new irregular sampling algorithm called *gaps* was

developed which was truly progressive, in that samples are generated one at a time. While there are similarities with FPS, *gaps* worked better on a wider range of images than FPS and with all reconstruction algorithms.

Both skewness-based sampling and *gaps* lead to blurry recovered images when using Multilevel B-Spline Approximation (MBA). Since at times there is no control over the placement of samples, new reconstruction methods have to be developed. This was done in Chapter 5 of this thesis, which introduced New Edge-Directed Multilevel B-Spline Approximation (NEDMBA) is introduced. This algorithm maintains the positive features of MBA, for example locality, while giving sharper edges in the reconstructions.

Finally, a new technique for digital image halftoning was introduced in Chapter 6. A point-process approach was followed, which leads to very fast halftoning once the threshold array is formed. A more efficient irregular sampling algorithm known as Faster Farthest Point Sampling, based on FPS was given in Chapter 3. A generalization of FFPS was used for a new halftoning algorithm called Farthest Point Halftoning, which gives halftones with fewer of the textures and regular structures evident in the previously existing algorithms.

Irregular sampling, scattered data interpolation and halftoning are important problems on which much previous work has been done. This thesis presents a state-of-the-art summary of the field, and contributes new algorithms; *gaps*, NEDMBA and Farthest Point Halftoning. This thesis can provide foundations for further research

so that better image compression and rendering become a reality.

7.2 Future Work

As mentioned in previous chapters, the sampling and reconstruction algorithms can be extended from gray-level to color images. In fact, it should even be possible to generalize Farthest Point Halftoning to color halftoning using similar ideas to [13, 42], where a different mask is used for each color plane.

While *gaps* is a very good and reasonably efficient progressive sampling algorithm, it may be improved by removing the requirement that the point map be transmitted, as is done with Farthest Point Sampling. This is practical since there are in general four relatively close samples used in the calculation of the gap magnitude of a pixel. A formula for the weight of a pixel similar to that used for adaptive Farthest Point Sampling can be used involving the intensities and locations of these samples. Faster Farthest Point Sampling has the attractive properties of fast execution time along with a lack of a point map, but the sample placement for a given number of samples is not as good as *gaps*.

In this thesis, new techniques have been presented for both irregular sampling and scattered data interpolation. The work could be extended to include a comparison of these techniques with respect to a coding scheme (e.g. with quantization and arithmetic coding) as in [11] in addition to comparing the performances of all

algorithms with JPEG at a given bitrate.

NEDMBA should be speeded up, and this could be done by using an interpolator other than New Edge Directed Interpolation (NEDI). This is because NEDI has to be called repeatedly (it only doubles the size of the image with each call) to create the zoomed image of the same size as the original. A more thorough analysis of why NEDMBA makes some edges sharper than others is needed, which may provide for an improvement in the appearance of all edges in reconstructions. A hybrid between adapted 4-Nearest Neighbor Interpolation (close to edges) and Multilevel B-Spline Approximation (in homogeneous areas) is another possibility that should be explored.

References

[1] High Performance Computing and Informatics Office, "Design Considerations for Medical Image Archive System", [Online document], 2000, [cited 2002 Oct. 6], Available: http://cmag.cit.nih.gov/file/repository.pdf.

[2] O. Egger, P. Fleury, T. Ebrahimi, and M. Kunt, "High-Performance Compression of Visual Information - A Tutorial Review - Part I: Still Pictures," *Proceedings of the IEEE*, vol. 87, pp. 976–1011, June 1999.

[3] Signal Processing Laboratory (LTS) at the Swiss Federal Institute of Technology/Lausanne (EPFL), "Very Low Bit Rate Coding of Visual Information - A Review", [Online document], 1995, [cited 2002 Oct. 8], Available: http://ltssg3.epfl.ch/publications/pdf/oe_iscas95.pdf.

[4] K.-K. Kwon, K.-N. Park, B.-J. Kim, S.-H. Lee, S.-G. Kwon, and K.-I. Lee, "Adaptive Postprocessing Algorithm for Reduction of Blocking Artifacts Using Wavelet Transform and NNF," in *Proc. of International Technical Conference*

on Circuits/Systems, Computers and Communications, Phuket, Thailand, July 2002.

[5] Joint Photographics Experts Group, "Welcome to JPEG", [Online document], 2002, [cited 2002 Oct. 11], Available: http://www.jpeg.org/JPEG2000.html.

[6] Shuo-yen Chao and Gregory Chew, "JPEG 2000 and Wavelet Compression", [Online document], 2000, [cited 2002 Oct. 12], Available: http://www-ise.stanford.edu/class/psych221/00/shuoyen/.

[7] S. Lee, G. Wolberg, and S. Y. Shin, "Scattered Data Interpolation with Multilevel B-Splines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, pp. 229–244, July-September 1997.

[8] J. A. Robinson and M. S. Ren, "Data-Dependent Sampling of Two-Dimensional Signals," *Multidimensional Systems and Signal Processing*, vol. 6, pp. 89–111, 1995.

[9] E.J.F. Broekers, "Regridding of Irregularly Spaced Seismic Data," Master's thesis, Centre for Technical Geoscience, Delft University of Technology, 1994.

[10] J. J. Helly, "Visualization of Ecological and Environmental Data," in *Data and information management in the ecological sciences: a resource guide* (W.K. Michener and J.H. Porter and S.G. Stafford, ed.), pp. 89–94, LTER Network Office, University of New Mexico, Albuquerque, NM, 1998.

[11] G. Ramponi and S. Carrato, "An Adaptive Irregular Sampling Algorithm and its Application to Image Coding," *Image and Vision Computing*, vol. 19, pp. 451–460, May 2001.

[12] Q. Yu, K. J. Parker, and M. Yao, "On Filter Techniques for Generating Blue Noise Mask," in *Proc. of IS&T 50th Annual Conference, Cambridge, MA*, May 1997.

[13] H. Kang, *Digital Color Halftoning.* SPIE Optical Engineering Press, 1999.

[14] Y. Eldar, M. Lindenbaum, M. Parat, and Y. Y. Zeevi, "The Farthest Point Strategy for Progressive Image Sampling," *IEEE Transactions on Image Processing*, vol. 6, pp. 1305–15, September 1997.

[15] A. S. Glassner, *Principles of Digital Image Synthesis, Volume 1.* Morgan Kaufmann Publishers, Inc., 1995.

[16] P. Green and R. Sibson, "Computing Dirichlet Tessellations in the Plane," *The Computer Journal*, vol. 21, pp. 168–73, May 1978.

[17] Zdravko Jeremic, "Voronoi Diagrams in Biology", [Online document], 1998 Jul. 15, [cited 2001 Mar. 1]: http://www.beloit.edu/~biology/zdravko/vor_paper.html.

[18] Silicon Graphics, Inc., "SGI - Services & Support: Standard Template Library Programmer's Guide", [Online document], 2001, [cited 2001 Feb. 27]: http://www.sgi.com/tech/stl/.

[19] P. G. Anderson, "Error Diffusion using Linear Pixel Shuffling," in *Proc. of IS&T Conference, PICS 2000, Portland, OR*, March 2000.

[20] R. Shahidi and C. Moloney, "The *Gaps* Method for Irregularly Sampling an Image," in *Proceedings of NECEC 2001, St. John's, NL*, November 2001.

[21] A. Mojsilovic and E. Soljanin, "Color Quantization and Processing by Fibonacci Lattices," *IEEE Transactions on Image Processing*, vol. 10, pp. 1712–25, November 2001.

[22] R. Ulichney, "Dithering with Blue Noise," *Proceedings of the IEEE*, vol. 76, pp. 56–79, January 1988.

[23] S. Hiller, O. Deussen, and A. Keller, "Tiled Blue Noise Samples," in *Proc. of Vision, Modeling and Visualization, Stuttgart, Germany*, pp. 265–271, November 2001.

[24] D. Shepard, "A Two Dimensional Interpolation Function for Irregularly Spaced Data," in *Proc. ACM 23rd National Conference, Washington, DC*, pp. 517–524, January 1968.

[25] G. Goodsell, "On Finding p-th Nearest Neighbours of Scattered Points in Two Dimensions for Small p," *Computer Aided Geometric Design*, vol. 17, pp. 387–392, April 2000.

[26] R. Shahidi and C. Moloney, "New Edge-Directed Multilevel B-Spline Approximation," in *Proc. of NECEC 2002, St. John's, NL*, November 2002.

[27] X. Li and M. Orchard, "New Edge Directed Intrepolation," in *Proc. of International Conference on Image Processing, Vancouver, BC*, September 2000.

[28] W. Lu and Y.-P. Tan, "Image Interpolation Based on Spatial and Spectral Correlations," in *Proc. of International Computer Symposium, Taiwan*, December 2000.

[29] T. Foley, "Weighted Bicubic Spline Interpolation to Rapidly Varying Data," *ACM Transactions on Graphics*, vol. 6, pp. 1–18, January 1987.

[30] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image Inpainting," in *Proc. of SIGGRAPH 2000, New Orleans, LA*, July 2000.

[31] S. Esedoglu and J. Shen, "Digital Inpainting based on the Mumford-Shah-Euler Image Model," Tech. Rep. 1812, Institute for Mathematics and its Applications, University of Minnesota, 2001.

[32] R. March, "Visual Reconstruction with Discontinuities Using Variational Methods," *Image Vision Comput.*, vol. 10, pp. 30–38, January/February 1992.

[33] T. Mitsa and K. J. Parker, "Digital Halftoning using a Blue Noise Mask," in *IEEE Conference on Acoustics, Speech, and Signal Processing, Toronto, ON*, pp. 2809–2812, May 1991.

[34] M. Yao and K. J. Parker, "Modified Approach to the Construction of a Blue Noise Mask," *Journal of Electronic Imaging*, vol. 3, pp. 92–97, January 1994.

[35] R. Shahidi and C. Moloney, "Digital Image Halftoning with Blue Noise," in *Proc. of NECEC 2002, St. John's, NL*, November 2002.

[36] R. Ulichney, "The Void-and-Cluster Method for Dither Array Generation," in *Proc. SPIE, Human Vision, Visual Processing and Digital Display IV, San Jose, CA*, vol. 1913, pp. 332–343, February 1993.

[37] K. E. Spaulding, R. L. Miller, and J. Schildkraut, "Methods for Generating Blue-Noise Dither Matrices for Digital Halftoning," *Journal of Electronic Imaging*, vol. 6, pp. 208–230, April 1994.

[38] R. Shahidi, C. Moloney, and G. Ramponi, "Farthest Point Halftoning," in *Proc. of 2003 IEEE - EURASIP Workshop on Nonlinear Signal and Image Processing, to be presented, Grado, Italy*, June 2003.

[39] P. W. Wong, "Entropy Constrained Halftoning using Multipath Tree Coding," *IEEE Transactions on Image Processing*, vol. 6, no. 11, pp. 1567–1579, 1997.

[40] Qing Yu, "Digital Multitoning Evaluation with a Human Visual Model", [Online document], 1997 Nov. 13, [cited 2002 Oct. 27], Available: http://henry.ee.rochester.edu:8080/users/qiyu/Paper/MT/mt97.html.

[41] T. Scheermesser and O. Bryngdahl, "Texture Metric of Halftone Images," *J. Opt. Soc. Am. A*, vol. 13, pp. 18–24, January 1996.

[42] Meng Yao, *Blue Noise Halftoning.* PhD thesis, University of Rochester, 1996.