ERROR TOLERANCE APPROACH FOR SIMILARITY SEARCH PROBLEMS

by

Hye-Yeon Cheong

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

May 2011

Dedication

To my loving Father,

To my beloved husband, Alexis Tourapis,

Acknowledgements

Foremost, I owe a great debt of gratitude to my advisor Prof. Antonio Ortega for his patience, inspiration, continuous support and encouragement, and friendship throughout my PhD study. I feel very lucky to have him as my advisor and I would have been lost without him.

I would also like to express my sincere gratitude to the rest of my thesis committee: Prof. Sandeep Gupta, Prof. Cyrus Shahabi, Prof. Jay Kuo, and Prof. Melvin A. Breuer, for their insightful comments, and hard questions.

I wish to thank my parents and sisters for their support and love. I would like to thank my mother-in-law, Litsa, for her patience, love, and invaluable advices.

Most importantly, words cannot describe how much i am thankful to my dearest husband, alexis, for his unconditional love and support.

Table of Contents

List Of Tables

List Of Figures

Abstract

As the system complexity increases and VLSI chip circuit becomes more highly condensed and integrated towards nano-scale, the requirement of 100% exact execution of designed operations and correctness for all transistors and interconnects is prohibitively expensive, if not impossible, to meet in practice. To deal with these problems, defect tolerance (DT) and fault tolerance (FT) techniques at the design and manufacturing stages have been widely studied and practiced. FT and DT techniques ideally try to mask all effects of faults and internal errors by exploiting and managing redundancies, which leads to more complex and costly system to achieve hopefully ideal or at least acceptable output quality at the expense of additional complexity cost.

On the other hand, a recently introduced error tolerance (ET) approach is an exercise of designing and testing systems cost-effectively by exploiting the advantages of a controlled relaxation of system level output quality precision requirement. The basic theme of ET approach is to allow erroneous output leading to imperceptible degree of system level quality degradation in order to simplify and optimize the circuit size and complexity, power consumption, costs as well as chip manufacturing yield rate. Motivation of ET approach is two-fold: By exploiting certain range of distortions/errors which lead to negligible impact on system level performance, i) a significant portion of manufactured dies with such minor imperfection of physical origin can be saved, thus increasing overall effective yield, and ii) considerable circuit simplification and high power efficiency is attainable by systematically and purposefully introducing such distortions/errors.

With a primary focus on similarity search problem within the scope of this ET concept, this thesis presents several methodologies to deal with the problems of system complexity and high vulnerability to hardware defects and fabrication process variability and consequently a lower yield rate.

Many real world similarity search problems present serious computational burden and remain a long standing challenge as they involve high dimensional search space, often largely varying database, and increasingly more dynamic and user-interactive search metrics. This leads to complexity (response time, circuit/power complexity) becoming more important criterion for a successful design. Thus, great potential benefit of ET concept can be reaped in such area.

First part of this thesis studies similarity search problem and presents a novel methodology, called *quantization based nearest-neighbor-preserving metric approximation algorithm*(*QNNM*), which leads to significant complexity reduction in search metric computation. Proposed algorithm exploits four observations: homogeneity of viewpoint property, concentration of extreme value distribution, NN-preserving not distance-preserving criterion, fixed query info during search process. Based on these, *QNNM* approximates original/benchmark metric by applying query-dependent non-uniform quantization directly on the dataset, which is designed to minimize the average NNS error, while achieving significantly lower complexity, e.g., typically 1-bit quantizer. We show how the optimal query adaptive quantizers minimizing NNS error can be designed "off-line" without prior knowledge of the query information to avoid the on-line overhead complexity and present an efficient and specifically tailored off-line optimization algorithm to find such optimal quantizer.

Three distinguishing characteristics of QNNM are statistical modeling of dataset, employing quantization within a metric, and query-adaptive metric, all of which allow QNNM to improve performance complexity trade-off significantly and provide robust result especially when the problem involves non-predefined or largely varying data set or metric function due to its intrinsic flexibility from query to query change.

With motion estimation (ME) application, QNNM with coarsest 1-bit quantizer per pixel (note that a quantizer for each pixel is different to exploit the correlation of input distribution) results in on average 0.01dB performance loss while reducing more than 70% to 98% metric computation cost.

Second part of the thesis, we present a complete analysis of the effect of interconnect faults in NNS metric computation circuit. We provided a model to capture the effect of any fault (or combination of multiple faults) on the matching metric. We then describe how these errors in metric computation lead to errors in the matching process. Our motivation was twofold. First, we used this analysis to predict the behavior of NNS algorithms and MMC architectures in the presence of faults. Second, this analysis is a required step towards developing efficient ET based acceptability decision strategies and can be used to simplify fault space, separate acceptable/unacceptable faults, and consequently simplify testing. Based on this model, we investigated the error tolerance behavior of NNS process in the presence of multiple hardware faults from both algorithmic and hardware architecture point of views by defining the characteristics of the search algorithm and hardware architecture that lead to increased error tolerance.

With ME application, our simulation showed that search algorithms satisfying error tolerant characteristics we defined perform up to $2.5dB$ higher than other search methods in the presence of fault, and they also exhibit significant complexity reduction (more than 99%) without having to compromise with the performance. Our simulation also showed that if error tolerant hardware architecture is used, the expected error due to a fault can be reduced by more than 95% and more than 99.2% of fault locations within matching metric computation circuits result in less than $0.01dB$ performance degradation.

Throughout this thesis, motion estimation process for video coding and vector quantization for image coding are tested as example applications to numerically evaluate and verify our proposed algorithms and modeling in actual practical application setting.

Chapter 1

Introduction

1.1 Relaxing the Requirement of Exact Solutions

A computer is an electronic device (hardware) designed to manipulate data according to prescribed logical operations (software). Computers, over half a century, have moved deeply towards becoming everyday commodities and are embedded in practically every aspect of our daily lives from laptops and PDAs to mobile phones, digital cameras, toys, and all sorts of electronic appliances/devices. As its range of applications and our dependence on such devices continue to increase, the complexity of computer hardware and software has also increased.

The progress of semiconductor manufacturing technology towards deep sub-micron feature sizes, e.g., sub-100 nanometer technology, has created a growing impact of hardware defects and fabrication process variability, which lead to reduction in both initial and mature yield rates[1] [34]. It is becoming economically more difficult to produce close to 100% correct circuits as they become more highly condensed and integrated towards nanoscale. The profitability of manufacturing such digital systems heavily depends on the fabrication yield since lower yield rates

[1]Yield is defined as the proportion of operational circuits to the total number fabricated.

can increase the chip manufacturing/verification/testing cost and delay their time-to-market. Computer software also has been increasingly complex and has become more vulnerable to system operation/programming errors (bugs).

We first introduce some related terminology. Manufacturing defect is any unplanned imperfection on the wafer such as scratches from wafer mishandling, or the result of undesired chemical and airborne particles deposited on the chip during manufacturing process [32]. But not all defects lead to faults, affecting the circuit operation. Causes of fault include design error, manufacturing defects, external disturbance, system misuse. An error is a manifestation of a fault in a system, where an element produces different logical value from its intended one. A fault in a system can occur without causing an error, which is referred to as latent fault. A failure occurs when a running system deviates from its designed behavior/function due to errors.

1.1.1 Fault and Defect Tolerant System

The requirement of 100% bug-free software and/or 100% correctness for transistors and interconnects (fault free) is prohibitively expensive, if not impossible, to meet in practice. To deal with these problems, defect tolerance (DT) and fault tolerance (FT) techniques at the design and manufacturing stages have been widely studied and practiced [32] [29].

Fault tolerance is the ability of a system to continue to provide correct service /operation of given task even in the presence of internal faults (either hardware failures or software errors), while defect tolerance refers to any circuit design/implementation that provides higher yield than the one without defect tolerance technique applied [6]. These approaches aim at masking all effects of defects or faults such that no error occurs at any output of a faulty system. Correct operation typically implies no error at any system output but certain types of errors might be acceptable depending on its severities. Although there exist many variations, the basic theme of all fault and defect tolerance techniques is to introduce/manage some form of redundancies/safeguards and extra functionalities in terms of the hardware, software, information, and/or time [33]. In other words, components that are susceptible to defects or faults are replicated, such that if a component fails, one or more of the non-failed replicas will continue to perform correct operations with no appreciable disruption. Note that both fault and defect tolerance approaches assume the ideal/exact output requirement and try to perform the exact action and produce the precise output. While it is very hard to devise absolutely foolproof, 100% reliable system, the more we strive to reduce the probability of error, the higher the cost. To put it differently, FT and DT techniques lead to more complex system to achieve hopefully the ideal or at least acceptable output so that the high quality reliability is attained at the expense of additional complexity cost.

For some applications such as medical devices or financial applications, the exact precise execution of designed operations without error is critical even if it comes with extra costs to safeguard and guarantee the output quality. However, for the enormous domain of other applications such as systems dealing with multimedia information, search engines, or power sensitive mobile devices, controlled relaxation of output quality precision is desirable and advantageous. This is because many applications of these kinds share a key characteristic of being able to tolerate certain types of errors at the system outputs within certain levels of severities. Furthermore, many complex problems and tasks entail approximation and estimation techniques at the algorithmic level to obtain the solution. Also note that most approximation algorithms proposed are not optimized in terms of circuit complexity (primarily optimized in time and space complexity with respect to the input size) and the measures they employ to specify what constitutes exact or optimal solution often does not accurately capture the notion of user quality. Therefore the restriction on exact precision in solution quality is, realistically speaking, an over-idealization.

For such systems and applications, a new application-oriented paradigm, *error tolerance* is recently introduced.

Definition of error tolerance [6] A circuit is error tolerant with respect to an application if (1) it contains defects that cause internal errors and might cause external errors, and (2) the system that incorporates this circuit produces acceptable results.

1.1.2 Error Tolerant System

In this thesis we particularly focus on this application-oriented error tolerance (ET) approach [6] which is fundamentally different from FT or DT. Within the scope of error tolerance concept, we develop in following chapters several methodologies which allow to deal with the problem of system complexity and high vulnerability to internal errors and faults in a different perspective.

The error tolerance approach assumes that the performance requirement itself is relaxed from the exact solution to an acceptable range of solutions and tries to exploit and maximize the benefit of this relaxed performance constraint to significantly simplify and optimize the circuit size and complexity, power consumption, costs as well as yield rate. Thus, the basic theme of ET approach is to sacrifice exact output quality to an acceptable range of output quality of system (results might be erroneous but by inappreciable or imperceptible degree of quality degradation) to make the system “simpler” and cost-effective, whereas FT and DT techniques

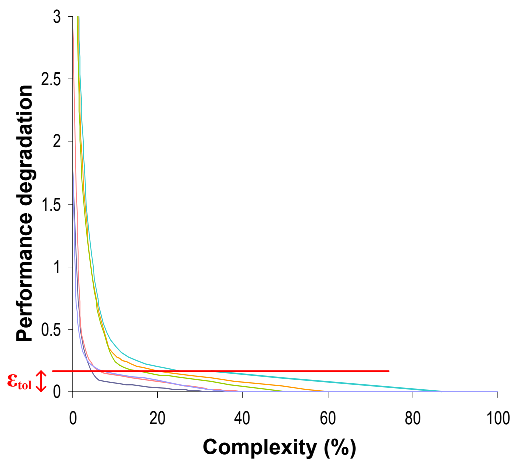


Figure 1.1: Simple illustration of error tolerance concept. Relaxing the requirement of exact solution/output to a small range ϵ_{tol} of near-optimal/ acceptable solution may dramatically reduce the cost. Cost can be interpreted as computational/ circuit complexity, power, manufacture, verification, testing cost and others. Curves or slopes of this trade-off relation can be controlled by different ET(approximation) algorithms or ET based design/ synthesis/ testing strategies. Curves shown in this figure are collected from different applications (motion estimation for video coding and vector quantization for image coding) when our proposed ET based algorithm ($QNNM$ in Chapter 2) is applied.

lead to more complex and costly system to achieve hopefully the ideal or at least acceptable output. Fig. 1.1 shows a simplified illustration of the error tolerance approach.

The importance of error tolerance approach also lies in its potential to significantly enhance the effective yield at the testing stage. Classical test techniques have certain deficiencies when applied to error-tolerant systems. First, they deal only with pass (perfect:fault free) versus fail (imperfect:faulty) whereas error tolerant testing makes use of the gray area between perfect and imperfect by further partitioning faulty dies into more categories. In addition, classical test techniques ignore error characteristics and, at most, relate errors to fault location. Error tolerance technique, on the other hand, does not focus only on fault location but rather on identifying the quantitative and qualitative implications of a defect. Finally,

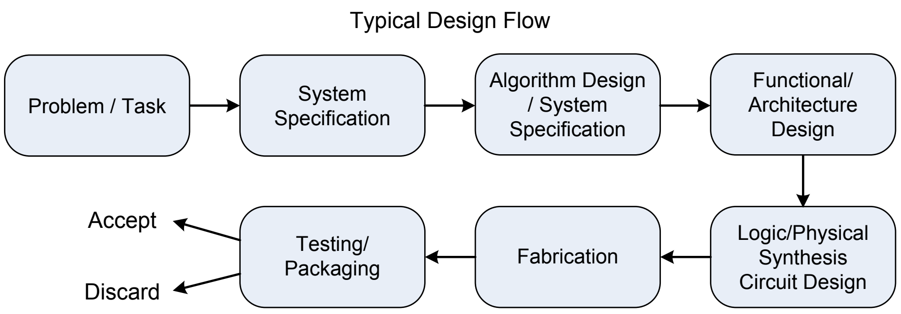

Typical Design Flow
Problem / Task
System Specification
Algorithm Design / System Specification
Functional/ Architecture Design
Logic/Physical Synthesis Circuit Design
Fabrication
Testing/ Packaging
Accept
Discard

Figure 1.2: Simplified illustration of a typical digital circuit design flow. ET approach, i.e., relaxed constraints on performance, may be applied to each design and synthesis phase.

classical tests are universal for a given part, whereas error tolerant testing might require application-specific tests [6].

Fig. 1.2 shows a typical digital circuit design flow consisting of a series of distinct activities that are typically (although not necessarily) performed in sequence. Error tolerance techniques can be applied to each phase from high level synthesis to logic level synthesis phase as well as testing phase. Refer to [1] [6] for further details on error tolerant computing and its related research.

In this thesis, we study one of the most fundamental problems called proximity problems in the context of error tolerance approach and present several ET based algorithms from the algorithmic level to hardware architecture and testing level. We primarily focus on the nearest neighbor search (NNS) problem, for it is central in a vast range of applications. Many real world NNS problems present serious computational burden and remain a long standing challenge as they involve high dimensional search space, often largely varying dataset, and increasingly more dynamic and user-interactive search metrics. We first present a novel ET based NNS algorithm called, *quantization based nearest-neighbor-preserving metric approximation algorithm* (*QNNM*) which provides a solution to significantly simplify the metric computation process while preserving the fidelity of the minimum distance ranking based on statistical modeling of data set (Chapter 2). In Chapter 3, we provide a model that estimates the impact of the manufacturing faults (single as well as multiple) within a NNS metric computation circuit on the search performance. We further derive the characteristics of the search algorithms and hardware architectures that lead to increased error tolerance property (i.e., reduced impact of a given fault on the output performance). Motion estimation process for video coding and vector quantization for data compression are tested as example applications for our study to verify our results numerically.

In the rest of this chapter, brief description of NNS problem (Section 1.2.1), two example applications, motion estimation (Section 1.2.2) and vector quantization (Section 1.2.3), and the overview and goal of this thesis (Section 1.3) are provided.

1.2 Proximity Problems

The proximity problem is a class of geometric problems in computational geometry which involve estimation of distances between geometric objects. Some example proximity problems on points include closest pair problem, diameter (furthest pair) problem, NNS, k-NNS, approximate NNS problems, minimum spanning tree, variants of clustering problems, range query, reverse query, batched query, and bi-chromatic problems.

Although in this thesis, techniques primarily focusing on the NNS problem are studied and developed, it is possible to apply and extend the same concept of our work to other proximity problems.

1.2.1 Nearest Neighbor Search (NNS) Problem

1.2.1.1 Definition of NNS problem

NNS problem we primarily consider in this thesis is simply defined as: given N points in metric space, with preprocessing allowed, how quickly can a nearest neighbor of a new given query point q be found?

That is, suppose U is a set and d is a distance measure (metric) on U such that $d : U \times U \rightarrow [0, \infty)$, taking pairs of objects in U and returning a nonnegative real number. Given a set $R \subset U$ of N objects and a query object $q \in U$ in a metric space $M = (U, d)$, NNS problem is to find efficiently the (approximately) nearest or most similar object in the set R to a query q $(min\,\{d(q, s)|s \in R\})$.

We chose NNS problem as a good example where the error tolerance property can be well exploited for the following reasons described below.

1.2.1.2 Central to a wide range of applications

This is among the most fundamental problems and a basic component of most proximity problems and it is also of major importance to a wide range of applications. NNS problem appears with different names (e.g., post office problem, proximity search, closest point search) in a diverse literature in the area of computational geometry, statistics [23], data analysis/mining, biology [44], information retrieval [45] [22], pattern recognition/classification [20], data compression [24], computer graphics [39], artificial intelligence [43]. Typically the features of each object of interest are represented as a point in a vector space $\mathcal{R}^D$ and the metric d is used to measure similarity between objects. Most optimization problems requiring finding a point/object minimizing or maximizing a given cost function can be also seen a special case of NNS problem.

1.2.1.3 Computational challenge

While it is of major importance to a wide range of applications, many real world search problems present serious computational burden and remain challenging as they take place in a very high dimensional search space (in the order of hundreds or thousands). Furthermore, some applications involve data set R arbitrarily varying from query to query (e.g., motion estimation) or even user controllable/interactive search metric d which may also change from query to query. There has been extensive research and a multitude of algorithms proposed to reduce computational complexity of these problems in terms of query time, storage space, and preprocessing complexity. But very little work has been done with respect to the efficiency of metric evaluation and circuit complexity.

NNS problem is also appropriate for the ASIC (application-specific integrated circuit) customized chip rather than for the general-purpose one, since it involves very heavy data access, regularized and repetitive computations in distance evaluation for every pair of objects. This adds extra advantage of applying application-specific error tolerance approach to NNS problem.

1.2.1.4 Tends to be tolerant to approximation

Pursuing the exact NNS solution becomes meaningful if a specification of what constitutes optimal solution, i.e., metric measure for NNS case, is accurate. Mathematically formulated metric function itself is often the approximation of certain quality of dissimilarity, e.g., perceptual similarity between images, and does not accurately capture the notion of user quality. Strictly speaking, even the physical distance can be seen as an approximation since for the ease of computation, the physical distance is represented with a single number that is the average of a series of distance measurement for each dimension, e.g., sum of measurements for each dimension, while more accurate description of proximity should be associated with a distribution of each dimensional distance. Meaning, even the nearest in physical distance may not be the true nearest.

Furthermore, the significance of "nearest" neighbor decreases as dimensionality increases while many real world problems are posed with high dimensionality [3]. This is because as dimensionality increases, distribution of random objects in terms of distance tends to be concentrated at the same distance.

Therefore finding the exact nearest neighbor may not be as meaningful as it appears to be and small distance difference based on a given metric seldom matters in practice. Also over years, researchers have shown that approximation allows simplification of algorithm in terms of computational complexity as well as hardware architectures while performance is often as good as the exact solution. Especially in high dimensional search space ($D > 10$) as is the case in many practical applications, almost all NNS algorithms allow a certain degree of approximation due to the exponentially increasing time and space requirement of exact NNS computation.

Furthermore, note that NN metric is only to identify/preserve NN information and not the distance itself. Thus, as long as nearest-neighbor can be identified, not all metric computation has to be done blindly to full precision and one could maintain fine precision only where it is needed for instance.

In the following section, we introduce motion estimation and compensation process for video coding and vector quantization process for data compression as example applications of NNS problem. These specific applications are chosen to evaluate our analytical results numerically on actual real world applications and to verify them experimentally.

1.2.2 Motion Estimation Process

One of the key factors of video compression efficiency is how well the temporal redundancy is exploited by motion compensated prediction. Motion compensation is a technique to represent a picture in terms of the transformation of a reference picture which is previously transmitted/stored in advance so the compression efficiency can be improved. Block motion compensation (BMC) is most commonly used, where the frames are partitioned into non-overlapping blocks of pixels (e.g., macroblocks of 16×16 pixels). Each block is predicted from a block of equal size in the reference pictures (previous and/or future frames) as illustrated in Fig. 1.3. The blocks are shifted to the position of the predicted block, where the shift is represented as a motion vector (MV). There are many variations from this simple form of motion estimation including variable block sizes, subpel accurate MV search, weighted prediction, etc. But the basic theme of motion estimation(ME) process is to determine motion vectors which maximize the efficiency of prediction or the compression efficiency. Thus, performance of ME process is critical to optimize coding in a rate distortion sense.

The ME process comprises a search strategy of the motion displacement offset (motion vector) for inter picture predictive coding and a matching metric computation. Practically, it is infeasible to use the ideal metric that computes the ultimate effect, including transform, quantization, and entropy coding, of all possible motion vectors (MVs) to select the best MV. Therefore, a simpler metric has been used instead, which searches for the MV that minimizes certain distance metric prior to residual coding from a certain set of candidate MVs. The search strategy of the ME process determines the set of candidate MVs and its refining process so that it tends to optimize the trade-off between complexity and performance. The

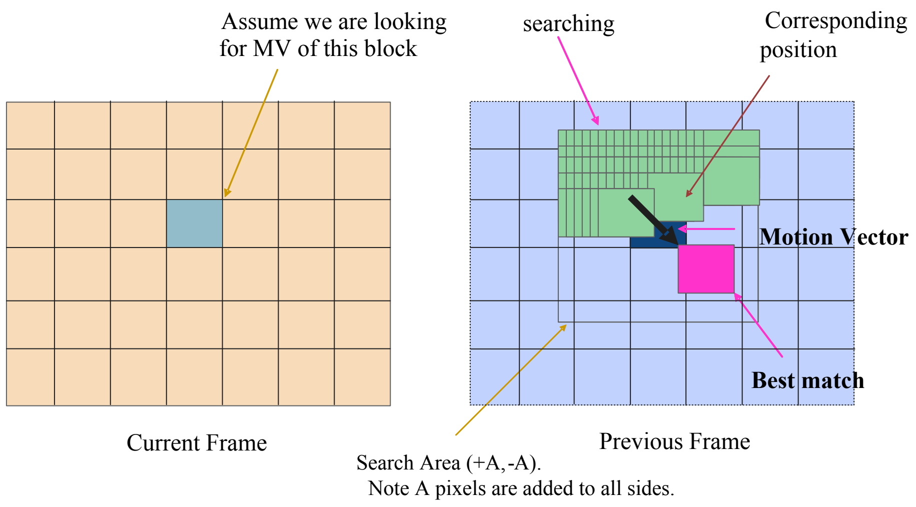


Figure 1.3: Motion estimation process. Current block as a query vector aims to selecting a motion vector that minimizes the distance metric.

cost/matching metric function which is also referred to as a distortion measure captures the prediction error; commonly l_1 or l_2 norms are used, i.e. Sum of Absolute Differences (SAD) or Sum of Squared Differences (SSD). After the encoder selects the MV that minimizes cost, it encodes the difference block (prediction residual) between the original and motion compensated blocks. Each residual block is transformed, quantized, and entropy coded. Fig. 1.3 shows a block diagram of motion compensated hybrid video coding scheme.

The motion estimation process for video coding is a particularly interesting application in the context of error tolerance due to its computational and memory intensive characteristics, the characteristic of compression itself (i.e., the lossy representation of uncompressed signals), and also the limited capacity of human visual perception ability.

ME can be seen as a NNS problem with a current block being a query object q and a set of candidate MVs from the reference pictures being a data set R. A

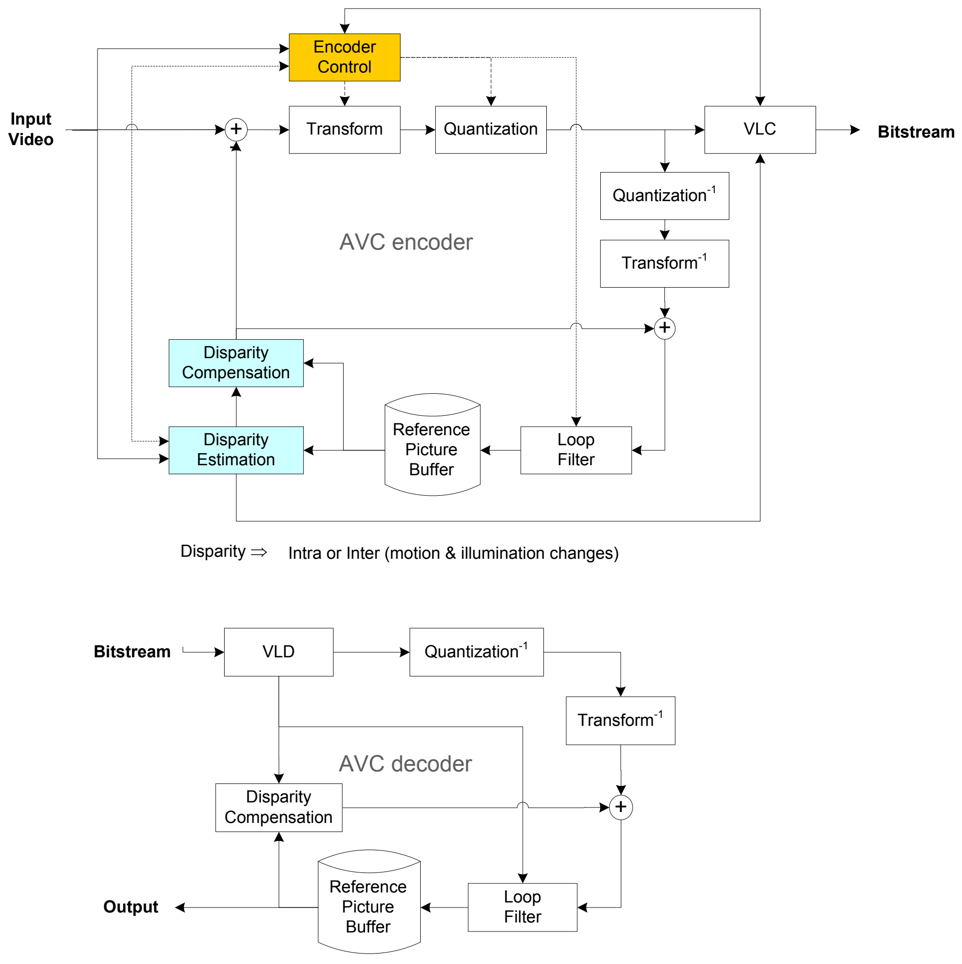


Figure 1.4: Block diagram of a hybrid coding scheme with motion compensating prediction.

difference with respect to the typical NNS problem is that a data set R is not fixed over queries and essentially R changes with q.

ME process tends to pose significant computational burden. For the simplest ME scheme using only 16×16 block size with a single reference frame and linear search with a search range ±32 could take more than 80% of the total encoding power of a video encoding system. Computational penalty of ME can be much worse if it includes other commonly used ME settings such as bi-predictive coding, multiple reference pictures, and variable block sizes.

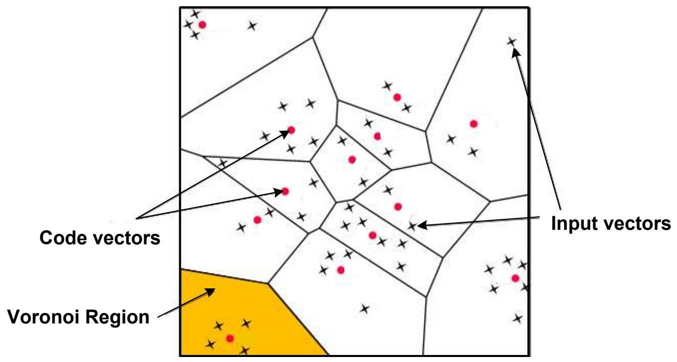
1.2.3 Vector Quantization for Data Compression

A vector quantizer encodes a multidimensional vector space into a finite set of values from a discrete subspace of lower dimension. A lower-space vector requires less storage space, so vector quantization (VQ) is often used for lossy compression of data such as image, video, audio or for voice recognition (statistical pattern recognition).

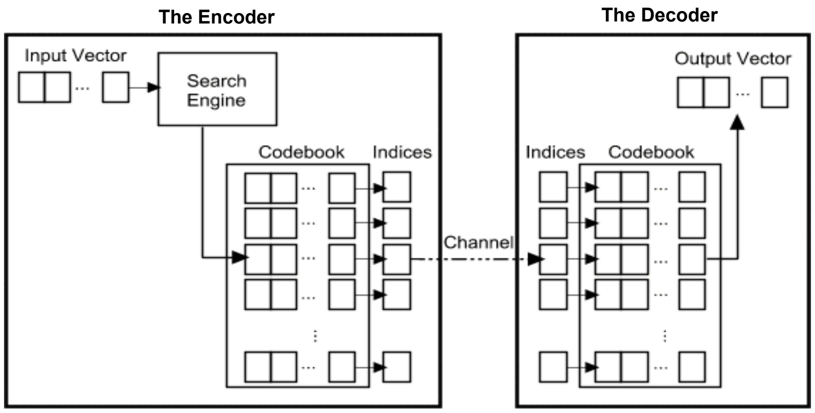
VQ maps D-dimensional vectors in the vector space $\mathcal{R}^D$ into a finite set of vectors $Y = \{\mathbf{y}_i : i = 1, 2, , N\}$. Each vector $\mathbf{y}_i$ is called a code vector and the set of all the code vectors, Y, is called a codebook. A vector space $\mathcal{R}^D$ is partitioned into N code cells (clusters) $C = \{C_i : i = 1, 2, , N\}$ of non-overlapping regions where each region C_i, called Voronoi region, is associated with each codeword/code vector, $\mathbf{y}_i$. Each Voronoi region C_i is defined by:

$$C_i = \{\mathbf{x} \in \mathcal{R}^D : \|\mathbf{x} - \mathbf{y}_i\| \leq \left\|\mathbf{x} - \mathbf{y}_j\right\|, \forall j \neq i\}$$

where VQ maps each input vector $\mathbf{x} \in C_i$ in code cell C_i to the code vector $\mathbf{y}_i$.



(a)



(b)

Figure 1.5: Vector quantization partitions a vector space $\mathcal{R}^D$ into a finite set of non-overlapping Voronoi regions. Each region is associated with a code vector and the set of all the code vectors is called a codebook. (a) illustrates an example design of a codebook for 2D vector space. (b) illustrates how each input vector is encoded with and decoded from the index of its nearest code vector.

VQ consists of two processes: (i) designing a codebook, i.e., generating the data set of NNS as a simple 2D codebook example illustrated in Fig. 1.5(a), and (ii) performing nearest neighbor/code vector search to a given input vector (a query of NNS) for encoding as illustrated in Fig. 1.5(b). There are several VQ algorithms which have different codebook design method but all of them perform exact nearest neighbor search. VQ performs NNS for each input vector (a query) to all code vectors (data set) and encodes it with the nearest code vector. A set of all code vectors can be seen as data set of NNS problem.

1.3 Contributions and Overview of the Thesis

In this thesis, with a primary focus on NNS problem, we propose to exploit error tolerance concept for proximity problems from algorithm level to hardware architecture design and testing level. The main theme throughout this thesis is to model and analyze the behavior of performance degradation with the range of faults/errors allowed or with the level of simplification made. Proposed modeling and algorithms throughout this thesis are numerically evaluated and verified by simulating using example applications: motion estimation process for video coding and vector quantization for data compression.

Quantization based nearest-neighbor-preserving metric approximation algorithm (*QNNM*) (Chapter 2) [11] [13]
First part of the thesis studies similarity search problem and presents a novel methodology, called *quantization based nearest-neighbor-preserving metric approximation (QNNM) algorithm*, which leads to significant complexity reduction in search metric computation. Proposed algorithm exploits four observations: (i)

homogeneity of viewpoint property, (ii) concentration of extreme value distribution, (iii) NN-preserving not distance-preserving criterion, and (iv) fixed query information during search process. Based on these, *QNNM* approximates original/benchmark metric by applying query-dependent non-uniform quantization directly on the dataset, which is designed to minimize the average NNS error, while achieving significantly lower complexity, e.g., typically 1-bit quantizer. It entails nonlinear sensitivity to distance such that finer precision is maintained only where it is needed/important (the region of expected nearest neighbors) while unlikely regions to be nearest neighbors are very coarsely represented. We show how the optimal query adaptive quantizers minimizing NNS error can be designed "off-line" without prior knowledge of the query information to avoid the on-line overhead complexity and present an efficient and specifically tailored off-line optimization algorithm to find such optimal quantizer. Three distinguishing characteristics of *QNNM* are statistical modeling of dataset, employing quantization within a metric, and query-adaptive metric, all of which allow *QNNM* to improve performance complexity trade-off significantly and provide robust result even when the problem involves non-predefined or largely varying data set or metric function. With motion estimation application, QNNM with coarsest 1-bit quantizer per pixel (note that a quantizer for each pixel is different to exploit the correlation of input distribution) results in on average 0.01dB performance loss while reducing more than 70% to 98% metric computation cost.

Manufacturing fault effect modeling and error tolerant designs for NNS problem (Chapter 3) [16] [9] [12]
Second part of the thesis, we present a complete analysis of the effect of interconnect faults in NNS metric computation circuit. We provided a model to capture

the effect of any fault (or combination of multiple faults) on the matching metric. We then describe how these errors in metric computation lead to errors in the matching process. Our motivation was twofold. First, we used this analysis to predict the behavior of NNS algorithms and MMC architectures in the presence of faults. Second, this analysis is a required step towards developing efficient ET based acceptability decision strategies and can be used to simplify fault space, separate acceptable/unacceptable faults, and consequently simplify testing. Based on this model, we investigated the error tolerance behavior of NNS process in the presence of multiple hardware faults from both algorithmic and hardware architecture point of views by defining the characteristics of the search algorithm and hardware architecture that lead to increased error tolerance. With ME application, our simulation showed that search algorithms satisfying error tolerant characteristics we defined perform up to $2.5dB$ higher than other search methods in the presence of fault, and they also exhibit significant complexity reduction (more than 99%) without having to compromise with the performance. Our simulation also showed that if error tolerant hardware architecture is used, the expected error due to a fault can be reduced by more than 95% and more than 99.2% of fault locations within matching metric computation circuits result in less than $0.01dB$ performance degradation.

Chapter 2

Approximation Algorithm for Nearest Neighbor Search Problem

In this chapter, we present a novel NNS metric approximation technique which maps the original metric space to a simple one in such a way that (approximate) nearest-neighbor (NN) is preserved, while reducing potential wasting of resources in computing high precision metric for unlikely solutions/points. This algorithm significantly reduces computational complexity while the penalty to be paid in performance is very small. This metric approximation is not fixed for all potential queries but adaptively adjusted at each query process, exploiting the information of query point and statistical modeling of data set to provide better performance complexity trade-off. Thus proposed method is intrinsically flexible from query to query and efficient with largely varying database/metric functions.

This NNS problem has a long history and we first summarize existing works and their approaches. We then provide certain intuitive interpretation of the notion of approximation and provide a different perspective which can be taken into account to further reduce computational complexity. Then we formulate the problem and present our proposed solution as well as experimental results.

2.1 Related Work

The NNS problem has been investigated in computer science for a long time and a multitude of algorithms have been presented, all of which focus on how to "*preprocess*" a given set of points R so as to efficiently answer queries assuming (i) that a data set R is fixed (or slightly varying) over queries and (ii) that a metric d is also fixed.

Approaches to NNS problem in general, can be broadly grouped into two classes : (i) reducing the subset of data to be examined (ability to discard large portion of data points during the search process), (ii) transforming the metric space.

The first class includes a variety of approaches that create efficient data structures by partitioning metric space and query execution algorithm (e.g., variants of k-d tree [47], R-trees, metric trees [52], ball-trees [41], similarity hashing [28]). The latter class includes metric/feature dimension reduction techniques, such as principal component analysis [26], latent semantic indexing [21], independent component analysis, multidimensional scaling and singular value decomposition using linear transforms (e.g., KLT, DFT, DCT, DWT). The latter class includes metric embedding techniques which provide ability to lower dimensions, transform general metric space to normed space, or map complex metric space to simpler one.

All these existing approaches focus on altering/preprocessing data set R so as to minimize volume (number of metric computation iteration) and size (e.g., dimensions) of examined data, using for example, data-restructuring [46] [48], filtering [40], sorting, sampling [36] [15], transforming [37], bit-truncating [25] [8], quantizing [35] etc, to ultimately reduce CPU and I/O costs for querying.

Although many among these algorithms provide good reduction in search complexity in low dimensionality, NNS problem with high dimensional and large collections of data set as is the case with many real-world problems, still remains a long standing challenge. Furthermore, since they are preprocessing-based techniques based on a fixed dataset and a metric function, they face serious challenges if NNS problem involves largely varying data set R changing significantly from query to query and/or variable/user-controllable search metric d. For applications requiring to solve such problem, most of existing methods simply use random or empirical sampling or application specific tailoring of data set according to the nature of data, which therefore is not generally applicable.

Along with any improvement achieved from these existing approaches, further significant complexity reduction is attainable by increasing the efficiency of metric computation. While most of works have been primarily focusing on designing algorithms which scale well with the database size as well as with the dimension, our proposed work focuses on error tolerant hardware implementation of metric computation for NNS search process. Our work can be seen as a technique that maps the original complex metric space to a significantly simpler metric space query-dependently, which is designed to minimize the average NNS error, while achieving significantly lower metric computation complexity. Furthermore, proposed method assumes neither a fixed dataset nor fixed metric function over queries, which enables efficient operation even with varying, user adaptive metric functions based on user preference and largely varying dataset.

Next section describes several conventional NNS performance evaluation measures and their pros and cons. We also introduce a different NNS performance measure and using this, we provide insight and intuition for why there might be more room for improvement in metric computation complexity.

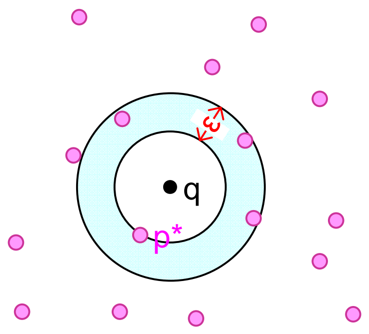


Figure 2.1: Illustration of approximate NNS problem ($\epsilon - NNS$), where q and $p*$ represent a query and its exact NN, respectively.

2.2 NNS Performance Evaluation Measure

Primary concerns of this problem are both qualitative and quantitative aspects of the solution: accuracy (how close the returned object is to the optimal object/solution) and complexity (the amount of resources required for the algorithm execution). Complexity of NNS algorithm typically refers to space complexity (memory requirement and preprocessing cost) and query time complexity (the number of metric evaluations to answer a query and computational complexity of metric evaluation). Complexity is in general compared as a function of the input size (size of a data set N and dimensionality D), typically represented by the worst case behavior. As to the accuracy, NNS problem can be either exact NNS or approximate NNS problem. Exact algorithms obviously reach an exact/optimum solution, while approximation algorithms find an approximate solution that is close enough to the true solution.

2.2.1 $\epsilon-$NNS Problem and Performance Measure

$\epsilon-$NNS problem

Most common approximate NNS problem is the $\epsilon-NNS$ problem (also called $c-$

NNS, where $c = 1 + \epsilon > 1$ is *approximation factor*) defined as follow: Given an error bound $\epsilon > 0$, we say that a point $\mathbf{r} \in R$ is an $\epsilon-$approximate NN of a query $\mathbf{q}$ if

$$d(\mathbf{r}, \mathbf{q}) \leq (1 + \epsilon) \cdot d(\mathbf{r}^*, \mathbf{q}),$$

where $\mathbf{r}^*$ is the exact NN of $\mathbf{q}$ (Fig. 2.1). In other words, an $\epsilon-NNS$ algorithm returns points whose distance from the query is no more than $(1 + \epsilon)$ times the distance of the true nearest neighbor, but, without any information on the actual rank of the distance of the returned points and also without any guarantee of returning any such point. In general as one decreases ϵ for better accuracy, the probability of returning an empty set increases.

Typical approach of evaluating/comparing $\epsilon-NNS$ algorithms is simply comparing complexity as a function of the input size and $\epsilon > 0$ as a notion of approximation measure providing guaranteed performance bound in a worst case point of view. However, in order to more accurately assess the quality of a returned set of an $\epsilon-NNS$ algorithm, beyond the performance bound measure ϵ, one could consider two complementary measures: *recall* [38] or *relative distance error (DE)* [53] defined as:

$$Recall = \frac{|R^* \cap R^A|}{|R^*|} \tag{2.1}$$

$$DE = \frac{d(\mathbf{r}^A, \mathbf{q}) - d(\mathbf{r}^*, \mathbf{q})}{d(\mathbf{r}^*, \mathbf{q})} = \frac{d(\mathbf{r}^A, \mathbf{q})}{d(\mathbf{r}^*, \mathbf{q})} - 1 \tag{2.2}$$

where R^* and R^A denote, respectively, a set of all relevant/qualifying points (all near neighbors within the range of distance $(1 + \epsilon) \cdot d(\mathbf{r}^*, \mathbf{q})$) and a returned set of points retrieved by the approximation algorithm of interest, while $| \cdot |$ represents the cardinality of a set. Similarly, $\mathbf{r}^*$ and $\mathbf{r}^A$ are the exact and retrieved (by the approximation algorithm) nearest neighbor of $\mathbf{q}$, respectively.

Recall measure represents the probability of approximate algorithm returning a qualifying point. This measure considers all points returned as being of equal importance. Relative distance error DE, on the other hand, quantifies the quality of a returned set but provides no indication on the probability of failing in returning any qualifying point. Note that DE measure is input data dependent.

In order to more accurately assess an improvement in complexity reduction due to a given algorithm, *improvement in efficiency (IE)* [53] measure can be used which relates the costs of the exact and approximated searches.

$$IE = \frac{cost(\mathbf{r}^*)}{cost(\mathbf{r}^A)} \tag{2.3}$$

where $cost(\mathbf{r})$ corresponds to the execution cost to retrieve $\mathbf{r}$.

2.2.2 Proposed $\bar{\epsilon}-$NNS Problem Setting

$\epsilon-$*NNS* problem constrains the quality of solution $\mathbf{r}$ to be within the worst performance bound $(1+\epsilon)\cdot d(\mathbf{r}^*, \mathbf{q})$ such that it only allows either to have a solution (that is close enough to the nearest neighbor based on ϵ) or to have nothing. Thus typically the performance of approximation algorithms has been analyzed using a worst case point of view. This worst case conditioning or worst case analysis provides a guaranteed performance safety and is useful in characterizing or proving whether the approximate solution is provably close to the optimal one independently from the input data distribution. However, it provides several drawbacks as well.

Worst case performance analysis generally does not serve as a practical predictive tool and is seldom met in practice: the empirical verification of approximation algorithm performance often shows large gap between actual expected performance and the worst case analytical bound. Moreover, for many useful practical algorithms involving probabilistic/randomized method or input data dependent processing, theoretical worst case analysis often becomes less meaningful since worst case can be very unlikely to happen in probability. Also for typical $\epsilon-NNS$ algorithms, reducing the worst case bound of solution quality (increasing the probability of having higher quality solution) leads to decreased recall rate (higher probability of returning an empty set) which is often input dependent.

Therefore in this thesis we consider approximate NNS problems without constraint on specific performance safety boundary. In other words, an approximate algorithm is expected to always return its own best result and its performance will be evaluated in probabilistic terms as the expected/average solution quality (closeness to the optimal one). We refer to it as $\bar{\epsilon}-$NNS problem.

This is more appropriate problem setting for our purpose, considering that our proposed work focuses on increasing the metric computation efficiency and not on creating a data structure and query process. To evaluate algorithms of interest in this setting, we only need to consider one NNS accuracy measure $\bar{\epsilon}$ measuring the average distance degradation introduced by approximate NNS algorithm relative to that of the benchmark algorithm (exact NN). $\bar{\epsilon}$, the solution quality as an expected value, is defined as:

$$\bar{\epsilon} = \frac{E(d(\mathbf{r}^A, \mathbf{q}) - d(\mathbf{r}^*, \mathbf{q}))}{E(d(\mathbf{r}^*, \mathbf{q}))} = \frac{E(d(\mathbf{r}^A, \mathbf{q}))}{E(d(\mathbf{r}^*, \mathbf{q}))} - 1 \tag{2.4}$$

where $E(\cdot)$ is the expectation operator over the input distribution. And similarly, complexity efficiency is measured as an expected value, $\overline{IE}$:

$$\overline{IE} = \frac{E(cost(\mathbf{r}^*))}{E(cost(\mathbf{r}^A))} \tag{2.5}$$

Although these measures provide more close numerical estimate to actual practical setting performance, reflecting average typical behavior in performance and complexity efficiency of given algorithm, they are intrinsically input data dependent. Therefore, to yield accurate results from a practical point of view, the input distribution needs to be as close as possible to the realistic distribution of the practical applications. Therefore, analysis on the performance behavior for different input characteristics may be needed as well for better evaluation.

Conventional $\epsilon-NNS$ and $\bar{\epsilon}-$NNS problem setting we propose to use can be briefly summarized as follow: Given the input distribution, $\epsilon-NNS$ algorithm performance can be evaluated using *Recall*, *relative distance error (DE)*, and data independent worst case bound ϵ,

$$d(\mathbf{r}^A, \mathbf{q}) \leq (1+\epsilon) \cdot d(\mathbf{r}^*, \mathbf{q}), \tag{2.6}$$

while the performance of proposed $\bar{\epsilon}-$NNS algorithm can be evaluated only with $\bar{\epsilon}$ (2.4).

$$E(d(\mathbf{r}^A, \mathbf{q})) = (1+\bar{\epsilon}) \cdot E(d(\mathbf{r}^*, \mathbf{q})) \tag{2.7}$$

2.2.3 Error Rate, Accumulation, Significance, and Variation

Particularly for the error tolerance approach, it is critical to select a good measure that quantifies the notion of error. Often *error rate, error accumulation, error significance* (these three are also known as RAS), and *error variation* measures are used. They are all input distribution dependent measure and quantify certain quality/behavior of errors under typical application specific environment. They

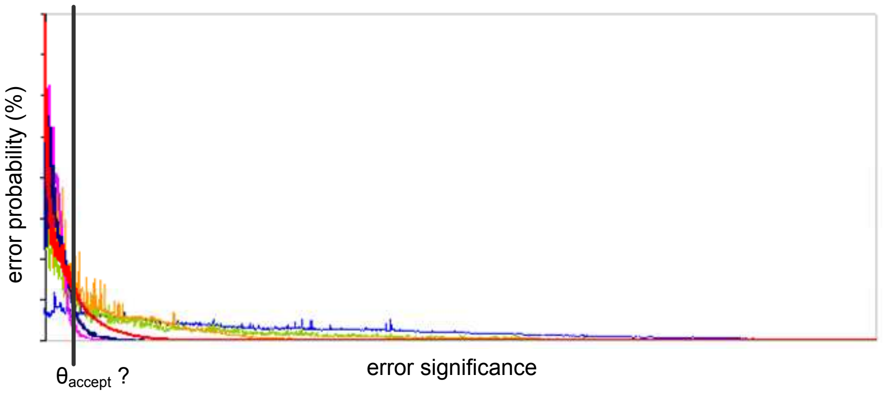


Figure 2.2: Illustration of error probability distribution over error-significance. Error-rate is defined as the sum of all non-zero error-significance probabilities. $\bar{\epsilon}$ measure is equal to the mean value of a given distribution. ET based acceptance curve would appear as a vertical line, determining a certain distribution to be classified acceptable or unacceptable depending on their mean value ($\bar{\epsilon}$).

describe more accurately the expected performance in practical setting unlike measures that are input distribution independent. Thus, such measures are consistent with our purpose of error tolerance approach.

As defined in [5], *error rate* is the fraction of results that are erroneous in a long sequence of output patterns under normal operating conditions. *Error significance* deals with the degree to which a piece of data is in error. *Error accumulation* (retention) deals with the change in error rate or error significance over time, which can be seen as the 'error propagation' measure. For example, in a feed-forward pipeline architecture, a defect probably produces a fixed error rate. Many finite state machines, such as a linear feedback shift register, lead to errors propagated or accumulated.

In the NNS problem setting, error rate refers to how often inexact solution is returned ($Pr(\mathbf{r}^* \neq \mathbf{r}^A)$) while error significance refers to how much additional quality degradation is introduced ($d(\mathbf{r}^A, \mathbf{q}) - d(\mathbf{r}^*, \mathbf{q})$). In our $\bar{\epsilon}-$NNS performance analysis, error rate is not very meaningful just as the worst case analysis is not a very good measure in practice. In fact, performance can be fully described by the error probability distribution over error significance as in Fig. 2.2. These distributions in this figure represent the NNS performance with different faulty NNS metric computation circuits. However, each distribution can be seen as the performance of an NNS approximation algorithm. Perfect system (e.g., fault-free chip, exact NNS algorithm) must have 100% error probability on zero error-significance. Error-rate is defined as the sum of all non-zero error-significance probabilities. Expected error significance, which is in fact non-normalized $\bar{\epsilon}$, is a good measure to describe such distribution with a single number, thus a good practical performance measure.

Note that if a system is sensitive to the temporal performance variation, variance of error distribution can be also taken into account for the acceptance decision. For example, when motion estimation for video coding is considered, if ME performance varies significantly from block to block, it could lead to unpleasant blocky artifact. However, in this thesis, we do not consider variance since the acceptable range of $\bar{\epsilon}$ is usually quite small (typically acceptable $\bar{\epsilon}$ should be constrained to "imperceptible /indistinguishable" degree of error to application users) and small $\bar{\epsilon}$ also assures small variance as well in practice.

2.3 Interpretation of NNS Approximation Using $\bar{\epsilon}$ Measure

The goal of this section is not to model all dependencies between different execution steps of all NNS algorithms but to provide a general form for algorithms so that we can obtain intuitive interpretation of approximation error.

2.3.1 NNS Problem Modeling

For this purpose, we model NNS problem in probabilistic terms using a distance distribution, instead of assuming specific fixed data set R. More specifically, given a metric space (U, d) where d defines distance between any pair of points in U and returns a nonnegative real number, $d : U \times U \rightarrow [0, \infty)$, we model a set of N points $R \subset U,\ R = \{r_1, r_2, \cdots, r_N\}$ as a set of N random samples from the distribution F_U over U. The relative distance distribution (RDD) of $d(\mathbf{r}, \mathbf{q})$ from a query $\mathbf{q} \in U$ with respect to any $\mathbf{r} \in U$ can be seen as a viewpoint taken from $\mathbf{q}$ towards a metric space U in terms of distance d, which is defined as:

$$F_q(\mathbf{x}) = Pr(d(\mathbf{r}, \mathbf{q}) \leq \mathbf{x}) \tag{2.8}$$

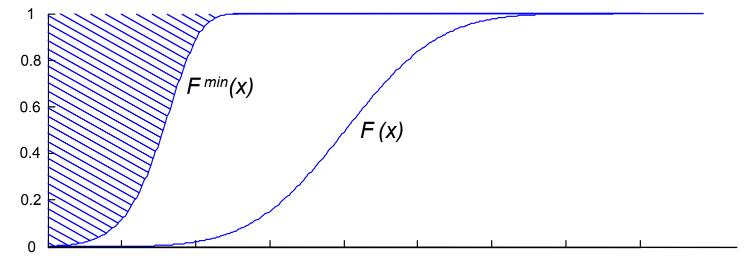
Similarly, a viewpoint taken from $\mathbf{q}$ towards its nearest neighbors in terms of d in U can be represented as the relative nearest-neighbors distance distribution (RNNDD) as:

$$F_q^{min}(\mathbf{x}) = Pr(d(\mathbf{r}^*, \mathbf{q}) \leq \mathbf{x}) = 1 - (1 - F_q(\mathbf{x}))^N \tag{2.9}$$

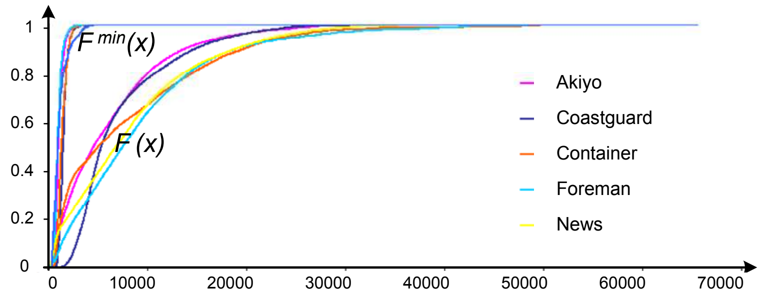
Fig. 2.3 illustrates examples of $F_q(\mathbf{x})$ and $F_q^{min}(\mathbf{x})$ for both simulated and actual data based cases.

2.3.2 NNS Algorithm Modeling

A linear search computes distances between $\mathbf{q}$ and all points $\mathbf{r} \in R$ and selects points in R which corresponds to the minimum distance. We denote by $NN(\mathbf{q})$ a set of minimum distance points with respect to $\mathbf{q}$ obtained using a linear search.



(a)



(b)

Figure 2.3: (a) Example illustration of simulated distance distribution, $F(x)$ and its corresponding NN-distance distribution, $F^{min}(x)$ for $|R| = 100$. Shaded area represents expected NN-distance $E(d(\mathbf{q}, NN(\mathbf{q})))$. (b) Illustration of both distance distribution, $F(x)$ and its corresponding NN-distance distribution, $F^{min}(x)$ based on the actual data collected from motion estimation process for video coding application using a linear search with $|R|$ approximately 4000. $F_q(\mathbf{x})$ and $F_q^{min}(\mathbf{x})$ are both averaged distributions. Five representatively different video sequences in terms of their distance distribution were chosen.

$$NN(\mathbf{q}) = \{\mathbf{r}^* \in R | \forall \mathbf{r} \in R \subset U, \mathbf{q} \in U : d(\mathbf{r}^*, \mathbf{q}) \leq d(\mathbf{r}, \mathbf{q})\} \tag{2.10}$$

We could also roughly characterize and model most NNS algorithms similarly. Any arbitrary NNS algorithm, which we refer to it as A from here on, can be modeled, in terms of its average behavior, as a linear search with different metric space (U_A, d_A) with a distribution F_{U_A}, thus consequently different R_A with size N_A and different d_A such that,

$$NN_A(\mathbf{q}) = \{\mathbf{r}^* \in R_A | \forall \mathbf{r} \in R_A \subset U_A, \mathbf{q}_A \in U_A : d_A(\mathbf{r}^*, \mathbf{q}_A) \leq d_A(\mathbf{r}, \mathbf{q}_A)\} \tag{2.11}$$

Similarly, we denote RDD and RNNDD of algorithm A with respect to $\mathbf{q}$ by $F_{q,A}(\mathbf{x})$ and $F_{q,A}^{min}(\mathbf{x})$.

For example, efficient NNS algorithms are likely to have distribution F_{U_A} concentrated near given $\mathbf{q}$ and $N_A < N$, which means algorithm A efficiently discarded most of points in R that are less likely to be $\mathbf{q}$'s nearest neighbors and selected R_A, a subset of R for actual comparison process.

2.3.3 NNS Algorithm Performance

The performance of NNS algorithm A can be measured by $\bar{\epsilon}$ as in 2.4:

$$\bar{\epsilon} = \frac{E(d(\mathbf{r}^A, \mathbf{q}) - d(\mathbf{r}^*, \mathbf{q}))}{E(d(\mathbf{r}^*, \mathbf{q}))}$$

For convenience in description, we use non-normalized $\bar{\epsilon}$ denoting it by simply $\overline{DE}$.

$$\overline{DE} = E(d(\mathbf{r}^A, \mathbf{q})) - E(d(\mathbf{r}^*, \mathbf{q})) \tag{2.12}$$

$\overline{DE}$ can be further written as:

$$\overline{DE} = \int_{R^+} \mu(\mathbf{x}) f_A(\mathbf{x}) dx - \int_{R^+} x f(\mathbf{x}) dx, \tag{2.13}$$

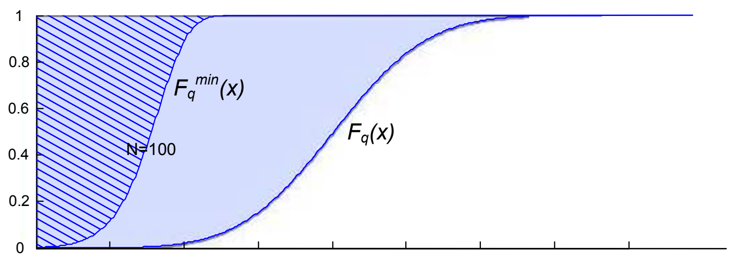
where $f(\mathbf{x})$ is a point probability or pdf (probability density function) form of $F_q^{min}(\mathbf{x})$, and analogously $f_A(\mathbf{x})$ is a pdf of $F_{q,A}^{min}(\mathbf{x})$. $\mu(\mathbf{x})$ represents the expected distance (in terms of the original distance measure d) of $\mathbf{q}$'s neighbors that have a given distance $\mathbf{x}$ according to the algorithm A's distance measure d_A.

$$\mu(x) = E(d(\mathbf{r}, \mathbf{q}) | d_A(\mathbf{r}, \mathbf{q}) = x, \forall \mathbf{r} \in R_A)$$

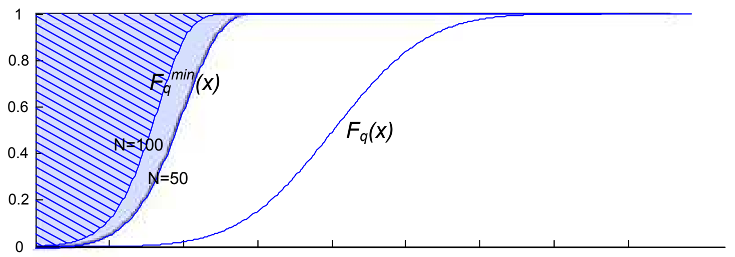
Given the input distributions, since $E(d(\mathbf{r}^*, \mathbf{q}))$ is fixed (it is equal to the shaded area in Fig. 2.3 (a)), the $E(d(\mathbf{r}^A, \mathbf{q}))$ term will determine the performance of algorithm A in terms of the solution quality. In other words, our goal is to find an algorithm which minimizes $E(d(\mathbf{r}^A, \mathbf{q}))$ while maximizing complexity reduction. Note that $\bar{\epsilon}$ or $\overline{DE}$ provide no information on the complexity efficiency which has to be considered separately.

2.3.4 Simple Case Study & Motivation

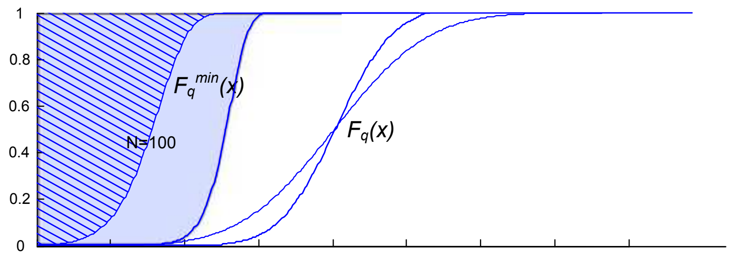
Fig. 2.4 is a graphical illustration of $\overline{DE}$ (where it can be seen to be the difference between the solid shaded area and the shaded area with hatching) for several simple NNS algorithms: (a) when algorithm A returns a randomly chosen point from R, (b) when A randomly subsamples R and performs a linear search and returns the minimum distance point, and (c) when A discards some information for distance metric computation (e.g., subsampling dimensionality) and performs a linear search. Solid shaded area and shaded area with hatching represent $E(d(\mathbf{r}^A, \mathbf{q}))$ and



(a)



(b)



(c)

Figure 2.4: Graphical illustration of $\overline{DE}$ ($= E(d(\mathbf{r}^A, \mathbf{q})) - E(d(\mathbf{r}^*, \mathbf{q}))$ = solid shaded area - shaded area with hatching) for several simple NNS algorithms. Three graphs illustrate the performance of three different algorithms which respectively (a) simply returns a randomly chosen point from R as its approximate NN point, (b) randomly subsamples R and performs a linear search to find the minimum distance point, and (c) discards some information for distance metric computation (e.g., subsampling dimensionality) and performs a linear search and return resulting NN point. Both (b) and (c) cases reduce the complexity by half (reducing the number of searching points N (b) and dimensionality D (c)).

$E(d(\mathbf{r}^*, \mathbf{q}))$, respectively. Case (a) can be seen as a worst case bound. While both (b) and (c) cases reduce the complexity by half (reducing the number of searching points N (b) and dimensionality D (c) by randomly subsampling), reduction in searching point impacts performance less than reduction in dimensionality of the metric function/data. However, due to the *curse of dimensionality*, efficient reduction in dimensionality usually helps reducing the size of searching points exponentially. Most existing NNS algorithms approach the problem by efficiently reducing either searching points N (e.g., using certain data structures) or dimensionality D (e.g., by performing certain transform process), or by a combination thereof. Well designed algorithms based on this approach can lead to minimal performance loss.

However, further significant reduction in complexity is attainable by reducing the metric computation resolution/precision instead of blindly computing each distance metric to full precision.

2.3.4.1 Nonlinear scaling on metric computation resolution

Fig. 2.5 provides intuitive idea of why significant resolution reduction may lead to insignificant performance loss. This can be seen as a problem of achieving the targeted NNS performance while minimizing the amount of resources (bits, number of quantization bins) required, by maintaining fine precision only where it is needed/important. Typical quantization is designed to compress the data while maintaining the necessary fidelity of the source data. However if we employ certain quantization process within the metric, the goal is not particularly in compressing the data but reducing the computational complexity while maintaining the fidelity of minimum distance ranking and not the fidelity of distance itself. Thus, our target metric needs to have a nonlinear sensitivity to distance, such that finer precision is assigned to the region of expected nearest neighbors and coarser resolution is

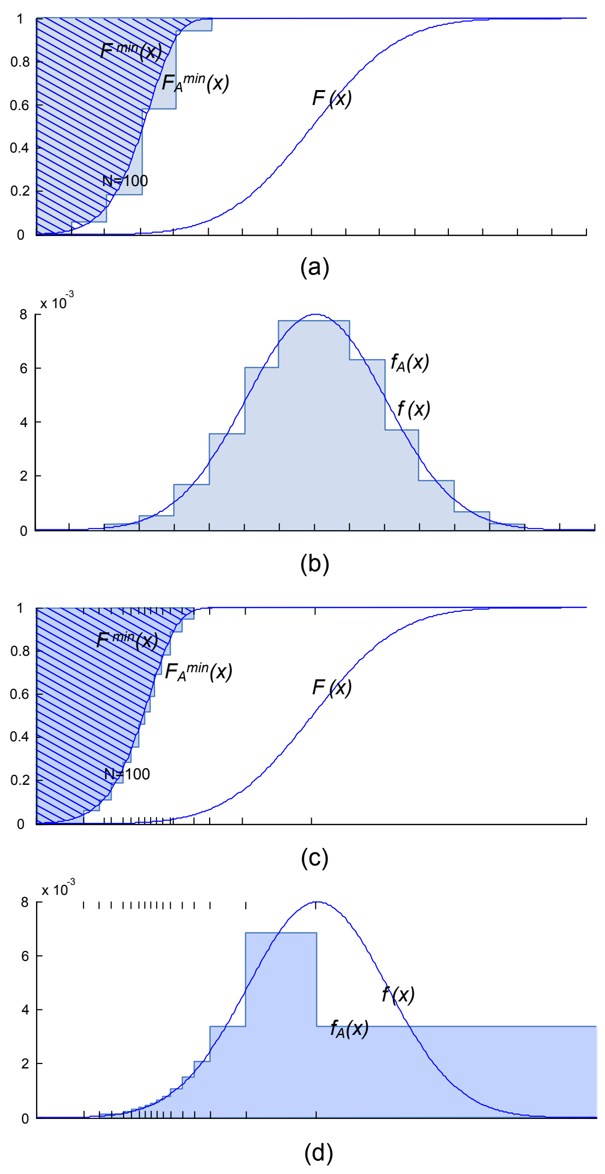


Figure 2.5: (a) Linear approximation of NN-distance distribution $F^{min}(x)$. (b) Linear approximation of distance distribution $F(x)$. (c) Nonlinear approximation of $F^{min}(x)$. (d) Nonlinear approximation of $F(x)$. Above four figures illustrate the impact on NNS performance when reducing the metric computation resolution/precision (staircase functions: 16 representation levels for these example graphs) instead of blindly computing each distance metric to full precision (solid blue line: e.g., 65536 representation levels if 16 bit-depth used).

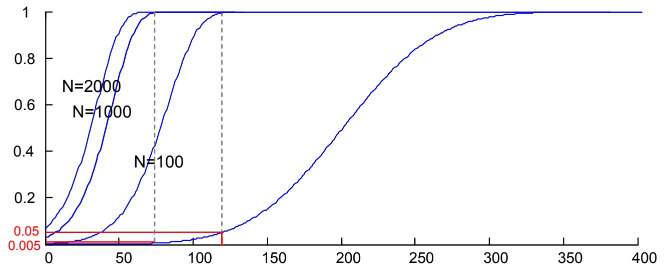


Figure 2.6: illustrates that most of critical regions we need to measure the distance accurately is the region belonging to less than 1% of a given RDD distribution (red box).

assigned to the rest of the regions. Fig. 2.5 (a) and (b) represent a technique which preserves the distance $F(x)$ well as in Fig. 2.5 (b) but results in higher error in NNS performance as in Fig. 2.5 (a). Fig. 2.5 (c) and (d) on the other hand is poor in preserving distances $F(x)$ as in Fig. 2.5 (d), while entailing less error in terms of the quality of approximate nearest neighbors (i.e., preserving $F^{min}(x)$ better) as in Fig. 2.5 (c).

2.3.4.2 Homogeneity of viewpoints

Discussion so far may raises an obvious question concerning to the statistical information that we are assumed to know for a fixed $\mathbf{q}$. The relative distance distribution $F_q(\mathbf{x})$ may vary for different viewpoint $\mathbf{q}$, which is called *viewpoint discrepancies*. This problem of *viewpoint discrepancies* has been studied [19] and it has been shown that the *viewpoint discrepancies* is quite insignificant for bounded random metric space, which is referred to as *homogeneity of viewpoints* property. Experiments performed with large text files also showed similar results. In other words, distance distributions measured with respect to different viewpoint $\mathbf{q}$, are very similar. However, note that for our purpose, even this assumption of high *homogeneity of viewpoints* is not necessary. Fig. 2.6 illustrates that the most critical region where we need to measure the distance accurately is the region belonging to less than 1% of whole RDD distribution (region in the red box). Considering that homogeneity of viewpoints "towards nearest-neighbors" is even higher, a technique developed based on the average statistics would behave quite well in general for different viewpoints **q**.

Fig. 2.3 (b) also provides real application information on the *homogeneity of viewpoints towards nearest-neighbors* for motion estimation application. Five representatively different video sequences in terms of RDD still produce high *homogeneity of viewpoints towards nearest-neighbors*. Fig. 2.3 (b) also shows that in general there is very limited region where high precision distance measure is needed for high NNS performance.

Following sections describe how this nonlinear scaling on metric computation precision can be exploited and how it can be designed to result in significant complexity reduction.

2.4 Query Adaptive Nearest Neighbor Preserving Metric Approximation Algorithm (QNNM)

2.4.1 Problem Formulation

Our goal is to find a new metric function d_{obj} that approximates the original or benchmark metric d in terms of preserving the fidelity of NNS while having significantly lower computational complexity than that of d. Any metric approximation approach can be formulated as $\psi : U \to U_\psi$ mapping the original metric space (U, d) into a simpler metric space (U_ψ, d_ψ) where NN search is performed with d_ψ metric.

If ψ is the same for all queries, this metric space mapping can be seen as a preprocessing (e.g., space transformation to reduce dimensionality) aiming at simplifying the metric space while preserving relative distance between objects. However, here our goal is to design a simple query-adaptive mapping $\psi_q : U \to U_\psi$

$$(U, d) \xrightarrow{\psi_q} (U_\psi, d_\psi) \tag{2.14}$$

that satisfies following three conditions:

1. Use the information of a given query location $\mathbf{q}$ which is constant over the searching process such that our objective metric d_{obj} can be designed as a function of only $\psi_q(\mathbf{r})$.
$$d_{obj}(\mathbf{q}, \mathbf{r}) = d_\psi(\psi_q(\mathbf{r})) \tag{2.15}$$

2. Reduce d_{obj} complexity by exploiting the statistical characteristics of NN (extreme value distribution of sample minimum F^{min} as in Fig. 2.3 where most of statistical information of NN is concentrated in a very narrow range) such that its resulting (U_ψ, d_ψ) is significantly simplified to preserve only NN and not the relative distances between objects.

3. Finding such query-dependent mapping ψ_q prior to each querying operation should impose insignificant overhead complexity, if any.

2.4.2 Observations & Characteristics of Proposed QNNM

Our proposed solution *QNNM* to this problem is based on the following four observations:

1. The *homogeneity of viewpoint* (HOV) property [19]: the distribution of data set R with respect to distance between two objects or nearest two objects tends to be homogeneous. In other words, if we statistically model a metric data set R in terms of its distances $d(\mathbf{q}, \mathbf{r})$ with respect to a given query $\mathbf{q}$ using the distance distribution $F(\mathbf{x}) = Pr(d(\mathbf{q}, \mathbf{r}) \leq \mathbf{x})$ or NN-distance distribution $F^{min}(\mathbf{x}) = Pr(d(\mathbf{q}, \mathbf{r}^*) \leq \mathbf{x})$, [19] shows that such distributions tend to be probabilistically very similar regardless of a query /viewpoint position.

2. As illustrated in Fig. 2.3, when performing NNS for different queries the distances $d(\mathbf{q}, \mathbf{r}^*)$ between a query vector and its best match (NN) tend to be concentrated in a very narrow range (*extreme value distribution of the sample minimum* $F^{min}(\mathbf{x})$).

3. The only goal of NNS metric is to identify NN or preserve the fidelity of the minimum distance ranking and not the distance itself.

4. The query vector is fixed during the entire search process.

Our proposed approach, motivated from these observations, has three distinguishing characteristics: (i) statistical modeling of data set R in terms of distance d, (ii) employing scalar quantization within a metric, and (iii) making a metric query-dependent.

Typically most of NNS related studies are based on the assumption of fixed data sets and a fixed distance metric d which are known in advance. However, many practical applications/situations, e.g., data streaming environments, involve largely varying data sets. For some applications such as motion estimation, more than half of data set changes from the current query to the following one. In such situations, preprocessing based algorithms lose their meaning because such assumptions consequently require to preprocess R partially or entirely all over again whenever there is a change in R or d.

The proposed algorithm, on the other hand, is based on the statistical characteristics of data set R (which is typically modeled using Gaussian distribution) in metric space (U, d) such that even if the search involves a largely varying data set R, no extra processing/updating is required due to the HOV property.

Furthermore, because of these characteristics of statistical modeling, the proposed method can support not only predefined but also ad-hoc or online metric queries. For example, user could control and select features of interest and their weights and assign each feature/dimension a different role for evaluating the similarity. Unfortunately most existing approaches construct whole preprocessed data structures based on a fixed dimensions and metric function. Our proposed method, on the other hand, is based on query adaptive metric which can maximize the use of query location information to simplify the metric as well as make the metric flexible and adaptable from query to query changes even when there exist variation of metric functions and/or dimensionality change, without having to rebuild the whole data structure or to perform transforms from scratch. Therefore, even when the metric d changes (e.g., change of $\mathbf{r}$ or weights w_j of weighted Minkowski metric $\sum_j w_j(q_j - r_j)^p$ especially when user-controllable), the proposed method does not require computing metric approximation from scratch again but only simple scaling of quantization thresholds.

Scalar quantization is chosen since it is computationally efficient, flexible to adapt to changed context (e.g., R, $\mathbf{q}$, d), but also convenient to introduce different sensitivity to different regions and dimensions to exploit the observation 2 and to reduce potential wasting of resources in computing high precision metric for unlikely solutions/points. Observation 2 allows these quantizers to be very coarse (e.g., 1 bit per dimension) leading to very low computational complexity without affecting overall NNS performance much. Our experimental results in Section 2.7.1, for instance, show negligible NNS performance degradation (average 0.01dB loss) when proposed method with optimized 1-bit/pixel quantizer is used as compared to ℓ_1 metric for motion estimation (ME) application for video coding.

The third characteristic of proposed metric approximation algorithm is that the metric measure itself is not fixed but changes with query data which is fixed during the entire search process. But more importantly, we show that the problem of finding the optimal *query-dependent* quantization parameters for the proposed metric can be formulated as an off-line optimization process based on observation 1, such that it only requires trivial overhead complexity of changing a metric for a new query data prior to the search operation.

In addition, note that our proposed QNNM algorithm can be used independently but also in parallel with most existing preprocessing based NNS algorithms. Because most preprocessing based algorithms try to prune and eliminate irrelevant data to a given query to narrow down the pool of data, which results in eventually different data set from query to query and therefore, it can be seen as a varying data set itself to further perform NNS within that archived data set. Also note that our approach (if optimally designed) automatically eliminate certain dimensions whose contribution to finding nearest-neighbor is insignificant. However our method cannot, for instance, transforms the basis/coordinate of the data to achieve dimensionality reduction. Therefore, if the user prefers, the data could be first preprocessed into a more suitable/compact domain to achieve better performance using, for example, principal component analysis [26], latent semantic indexing [21], independent component analysis, multidimensional scaling and singular value decomposition using linear transforms (e.g., KLT, DFT, DCT, DWT).

Finally, our proposed approach can be also used for k-nearest neighbor search or orthogonal range search where quantization itself is user specified input. If one insists on finding the exact nearest-neighbor, this approach can be also used as a preliminary filtering step to filter out unlikely candidates and then refinement process can be performed within the remaining set.

The details of proposed *QNNM* algorithm is described in following sections.

2.4.3 Basic Structure of Proposed QNNM

There are three assumptions under which proposed *QNNM* is developed:

1. D-dimensional vector space $U = \mathcal{R}^D$,

2. There is no cross-interference between dimensions in original metric d. In other words, each dimensional dissimilarity is measured in an isolated fashion and then averaged/combined together (refer to Table. 2.1). General metric function structure d can be written as:

$$d(\mathbf{q}, \mathbf{r}) = \sum_{j=1}^{D} d_j(q_j, r_j), \quad \mathbf{r} \in U \tag{2.16}$$

   This metric computation structure comprises two basic processes: i) the distance computation in each dimension (we will refer to it as *dimension-distance* $d_j(q_j, r_j)$), and ii) the summation/combination of all such dimension-distances.

3. The performance of NNS algorithm in terms of its accuracy is evaluated using the expected solution quality, $\bar{\epsilon}$ (the degree of average solution quality

Generalized metric structure of interest	$d(q,r)=\sum_{j=1}^{D} d_j(q_j,r_j)$
Manhattan	$d(q,r)=\sum_{j=1}^{D} \lvert q_j - r_j \rvert$
Euclidean	$d(q,r)=\left(\sum_{j=1}^{D} \lvert q_j - r_j \rvert^2\right)^{1/2}$
Minkowski	$d(q,r)=\left(\sum_{j=1}^{D} \lvert q_j - r_j \rvert^p\right)^{1/p} \qquad p\in R_+$
Weighted Minkowski	$d(q,r)=\left(\sum_{j=1}^{D} w_j \lvert q_j - r_j \rvert^p\right)^{1/p}$
Generalized Minkowski	$d(q,r)=\left(\sum_{j=1}^{D} c_j \lvert q_j - r_j \rvert^p\right)^{1/p} \qquad c_j > 0,\ \sum_{j=1}^{D} c_j = 1$
Camberra	$d(q,r)=\sum_{j=1}^{D} \frac{\lvert q_j - r_j \rvert}{\lvert q_j + r_j \rvert}$
Inner product (similarity)	$d(q,r)=\sum_{j=1}^{D} q_j \cdot r_j$

Table 2.1: Generalized metric function structure of our interest (average/sum of each dimensional distance measurements) and corresponding metric function examples.

degradation introduced by the algorithm relative to that of benchmark (full search) algorithm).

$$\bar{\epsilon} = E_q \left(\frac{d(\mathbf{q}, \mathbf{r}^*_\psi(\mathbf{q})) - d(\mathbf{q}, \mathbf{r}^*(\mathbf{q}))}{d(\mathbf{q}, \mathbf{r}^*(\mathbf{q}))} \right) \tag{2.17}$$

where

$$\mathbf{r}^* = \{\mathbf{r}' \in R | \forall \mathbf{r} \in R \subset U, \mathbf{q} \in U : d(\mathbf{q}, \mathbf{r}') \leq d(\mathbf{q}, \mathbf{r})\}$$

$$\mathbf{r}^*_\psi = \{\mathbf{r}' \in R | \forall \mathbf{r} \in R \subset U, \mathbf{q} \in U : d_{obj}(\mathbf{q}, \mathbf{r}') \leq d_{obj}(\mathbf{q}, \mathbf{r})\}$$

Our proposed approach simplifies the metric computation by approximating a given metric using quantization process. We will first provide a brief description of quantization and then proceed to describe our solution.

2.4.3.1 Quantization

Quantization maps a sequence of continuous or discrete vectors into a digital sequence such that less bit-rate can be used to represent the information while maintaining whatever necessary fidelity of the data. Scalar quantizer simply divides 1-dimensional space into a set of non-overlapping intervals or cells $S = \{s_i; s_i = [\theta_i, \theta_{i+1})\}$ that cover all possible values. $\{\theta_i\}$ is a set of thresholds.

A quantizer in general consists of two mappings: a forward quantizer partitions input data space into disjoint and exhaustive cells and assigns to each cell s_i a mapping symbol b_i in some mapping symbol set B. Inverse quantizer assigns to each mapping symbol b_i a reproduction value r_i. Thus, any input that falls into one cell s_i is quantized to s_i's corresponding mapping symbol b_i, which is in turn inverse quantized to its reproduction value r_i. converting input data sequence into a set of mapping symbol is to compress input data into fewer bits while reproduction value is to reproduce the expected original vector from the mapping symbols.

The goal of a typical quantizer is to produce the best reproduction of input data given a bit-rate budget. In other words, a quantizer is designed (i.e., determining a set of thresholds, mapping symbols, and reproduction values) typically to minimize reproduction error/distortion (e.g., mean square error between data points and their representatives).

A forward quantizer function **q** which compresses the input data **x** and returns corresponding mapping symbol, can be expressed as:

$$Q(\mathbf{x}) = \sum_i b_i 1_{s_i}(\mathbf{x}) \tag{2.18}$$

while an inverse quantizer function Q^{-1} which returns the reproduction value r_i from a mapping symbol b_i, can be similarly expressed as:

$$Q^{-1}(\mathbf{x}) = \sum_i r_i 1_{b_i}(Q(\mathbf{x})) = \widehat{x} \tag{2.19}$$

where $\widehat{x}$ represents a reproduction value of **x**.

2.4.3.2 Non-uniform scalar quantization within the QNNM metric

The proposed technique is illustrated in Fig. 2.7. Our proposed metric space mapping function $\psi_q : U \to \widehat{U}$ is a vector of scalar quantization functions

$$\psi_q = (\psi_{q1}, \psi_{q2}, \cdots, \psi_{qD}) \tag{2.20}$$

where each ψ_{qj} is applied independently on j-th dimension r_j of $\mathbf{r} = (r_1, r_2, \cdots, r_D)$ $\in R$, such that

$$\psi_q(\mathbf{r}) = (\psi_{q1}(r_1), \psi_{q2}(r_2), \cdots, \psi_{qD}(r_D))$$

where $\mathbf{r} \in U$ and $\psi_q(\mathbf{r}) \in \widehat{U}$. Each ψ_{qj} is a non-uniform scalar quantizer chosen based on the query data and dataset R. Each quantizer ψ_{qj} may have different number of quantization levels and threshold values for different j.

This mapping function ψ_q divides the metric space U into a set of disjoint and exhaustive hyper-rectangular cells as shown in Fig. 2.7 (b) and assigns each cell

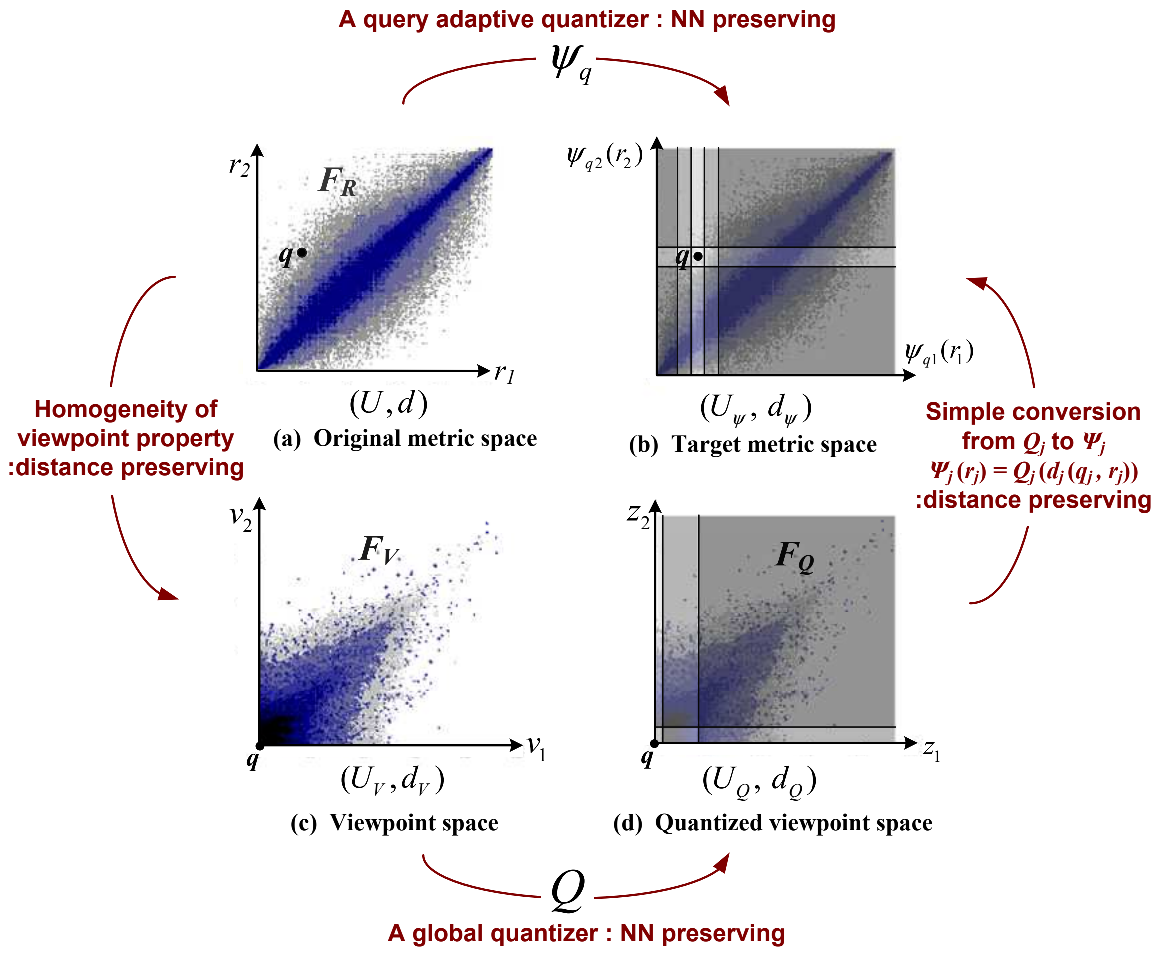

A query adaptive quantizer : NN preserving
ψ_q
r_2
F_R
q
r_1
(U, d)
(a) Original metric space
$\psi_{q2}(r_2)$
q
$\psi_{q1}(r_1)$
(U_ψ, d_ψ)
(b) Target metric space
Homogeneity of viewpoint property :distance preserving
Simple conversion from Q_j to Ψ_j
$\Psi_j(r_j) = Q_j(d_j(q_j, r_j))$
:distance preserving
v_2
F_V
v_1
q
(U_V, d_V)
(c) Viewpoint space
z_2
F_Q
z_1
q
(U_Q, d_Q)
(d) Quantized viewpoint space
Q
A global quantizer : NN preserving

Figure 2.7: Two dimensional example illustration of proposed $QNNM$ algorithm and the relation between ψ_q and $\mathbf{q}$ and their related spaces. F_R, F_V, and F_Q indicate the distributions of data set R represented in original (U, d), viewpoint (U_V, d_V), and quantized viewpoint space (U_Q, d_Q), respectively. This shows how ψ_q can be obtained from $\mathbf{q}$ such that overhead computation of finding ψ_q prior to each querying can be avoided.

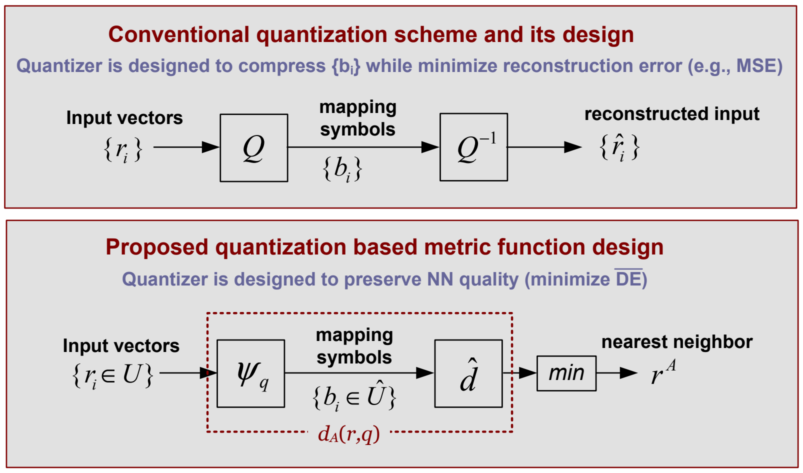

Conventional quantization scheme and its design
Quantizer is designed to compress {b_i} while minimize reconstruction error (e.g., MSE)
Input vectors
$\{r_i\}$
Q
mapping symbols
$\{b_i\}$
Q^{-1}
reconstructed input
$\{\hat{r}_i\}$
Proposed quantization based metric function design
Quantizer is designed to preserve NN quality (minimize $\overline{DE}$)
Input vectors
$\{r_i \in U\}$
ψ_q
mapping symbols
$\{b_i \in \hat{U}\}$
$\hat{d}$
min
nearest neighbor
r^A
$d_A(r,q)$

Figure 2.8: Comparison of conventional quantization scheme and our proposed quantization based metric function design.

with a cell index vector $\psi_q(\mathbf{r})$ (a vector of quantization integer mapping values). All data points $\mathbf{r} \in R$ need to be mapped to one of these cells. Based on their corresponding cell index $\psi_q(\mathbf{r})$, new distance to a query $d_{obj} = d_\psi(\psi_q(\mathbf{r}))$ (e.g., Fig. 2.9 (b)) is computed and a nearest distance point $\mathbf{r}^*_\psi$ in terms of a new metric is returned.

Note that our proposed quantization based mapping function has certain differences from typical quantization as illustrated in Fig. 2.8. The goal of our quantization based mapping function is not in minimizing input data reproduction error given a fixed bit-rate budget, but in minimizing $\bar{\epsilon}$, the average NNS performance loss given a fixed complexity budget.

Conventionally *rate* is defined as the average number of bits per source vector required to describe its corresponding mapping symbol. However we define *rate* as the cardinality of a set of mapping symbols B since the complexity of mapping function ψ_q increases proportionally with the size of B. Note that it is possible for multiple different cells to have the same mapping symbol.

2.4.3.3 Quantization-based metric design conditions

The design of (U_ψ, d_ψ) consists of determining d_ψ metric and a quantization based mapping function ψ_q (2.20). ψ_q is a set of scalar quantizers $\{\psi_{qj}\}_j$, each ψ_{qj} of which includes three sets of parameters; the number of quantization levels, a set of quantization thresholds, and a set of mapping symbols. The number of quantization levels and quantization thresholds determines how to partition a original metric space (U, d) into a set of hyper-rectangulars while quantization mapping symbols and d_ψ determines the ranking of each hyper-rectangular in terms of the probability of having NN. The whole design of these parameters should be chosen so as to maximize NNS performance in both search accuracy ($\bar{\epsilon}$) and computational complexity.

During similarity searching process, a query $\mathbf{q}$ is fixed and all metric computations are done to measure distances between $\mathbf{q}$ and other data points. By exploiting this fact, $d_\psi(\mathbf{q}, \mathbf{r})$ can be simply a function of only $\mathbf{r}$. Due to the second constraint of no cross-interference among dimensions in metric functions d, ψ_q can be designed in such a way that d can be reduced to d_ψ which can be a simply sum of vector elements:

$$d_\psi(\mathbf{q}, \mathbf{x}) = \sum_{j=1}^{D} x_j \tag{2.21}$$

or d_ψ can be even simpler bitwise OR operator of vector elements especially when ψ_q is 1-bit quantizer:

$$d_\psi(\mathbf{q}, \mathbf{x}) = \vee x_j \tag{2.22}$$

Note that mapping symbols of ψ_q can be designed more intelligently than simple consecutive numbers to simplify d_ψ complexity to bitwise operators.

If summation is used as d_ψ as in (2.21), d_ψ can be implemented with a simple D-leaf binary adder tree. d_{obj} which is our ultimate goal to find, can be represented as:

$$d(\mathbf{q}, \mathbf{x}) \cong d_{obj}(\mathbf{q}, \mathbf{x}) = d_\psi(\psi_q(\mathbf{q}), \psi_q(\mathbf{r})) = \sum_{j=1}^{D} \psi_q(\mathbf{r})_j = \sum_{j=1}^{D} \psi_{qj}(r_j) \tag{2.23}$$

In other words, any original/benchmark metric d satisfying $d(\mathbf{q}, \mathbf{r}) = \sum_{j=1}^{D} d_j(q_j, r_j)$ structure can be reduced to $d_{obj}(\mathbf{q}, \mathbf{r}) = \sum_{j=1}^{D} \psi_{qj}(r_j)$ metric if scalar quantizer set ψ_q is optimally designed in such a way that nearest-neighbor found based on d_{obj} metric is very close to that found based on original d metric (i.e., small $\bar{\epsilon}$).

Note that this metric d_ψ obviously violates the classical notion of metric (symmetry and triangle inequality conditions) since it only defines distance from a point to a fixed $\mathbf{q}$. Thus, this is only meaningful for similarity searching purpose.

Since our goal is *not* to reproduce or approximate the input point $\mathbf{r}$ but to simplify a new metric space U_ψ and d_ψ, mapping symbol set can be simply a set of consecutive integers beginning with 0 instead of, for example, centroid of corresponding quantization bin.

The benefit of compressing $\mathbf{r}$ into fewer bits for each dimension is to reduce the input bit size to d_ψ computation such that the binary adder tree of d_ψ (assuming d_ψ is summation operator) can be simplified significantly.

As mentioned earlier, however, mapping symbols can be also designed to simplify d_ψ into bitwise operator. Furthermore, it can be also designed to improve NNS performance (reduce $\bar{\epsilon}$) in a way to control the slope of d_{obj} metric surface depending on the input distribution of data set R.

2.4.3.4 Avoiding the overhead complexity

Given R and $\mathbf{q}$, ψ_q and d_ψ need to be designed to minimize both complexity and average NNS error $\bar{\epsilon}$ (2.17). However, finding the optimal query-dependent ψ_q satisfying such conditions prior to each querying operation would impose significant overhead costs. This can be avoided based on the *homogeneity of viewpoint* property studied in [19].

This allows to consider a *viewpoint space* (U_V, d_V) (e.g., Fig. 2.7(c)), where we denote $\mathbf{v}$ the vector of dimensional distances $d_j(q_j, r_j)$ (2.16) between a query point and a search point:

$$\mathbf{v} := \vec{d}(\mathbf{q}, \mathbf{r}) \in U_V. \tag{2.24}$$

where we define $\vec{d}(\cdot)$ operator as:

$$\vec{d}(\mathbf{x}, \mathbf{y}) := (d_1(x_1, y_1), d_2(x_2, y_2), \cdots, d_D(x_D, y_D)) \tag{2.25}$$

Then, under the assumption of viewpoint homogeneity (which makes distance distribution on viewpoint space query-independent), we can generate off-line statistics over multiple queries and model a dataset by the overall distance distribution F_V of $\mathbf{v}$ in U_V. Or if the distribution f_R of data set R can be fitted with standard distribution such as multivariate normal, the distribution f_V can be easily computed from cross-correlation of f_R and f_q (the distribution of queries) which is discussed in detail in Section 2.5. F_V of $\mathbf{v}$ in viewpoint space is the distribution of $\vec{d}(\mathbf{r}_1, \mathbf{r}_2)$ where $\mathbf{r}_1$ and $\mathbf{r}_2$ are i.i.d. with data set distribution f_R.

$$F_V(\mathbf{x}) = Pr(\mathbf{v} \leq \mathbf{x}) = Pr(\vec{d}(\mathbf{r}_1, \mathbf{r}_2) \leq \mathbf{x}), \quad \mathbf{r}_1, \mathbf{r}_2 \sim f_R \tag{2.26}$$

If the distribution of queries f_q is different from that of data set f_R, then

$$F_V(\mathbf{x}) = Pr(\mathbf{v} \le \mathbf{x}) = Pr(\vec{d}(\mathbf{q}, \mathbf{r}) \le \mathbf{x}), \quad \mathbf{q} \sim f_q, \mathbf{r} \sim f_R \tag{2.27}$$

where F_V represents the probability that there exist objects whose distance $\mathbf{v}$ to a given arbitrary query is smaller than $\mathbf{x}$.

Given this query-independent F_V model obtained, instead of directly finding $\psi_q : U \rightarrow U_\psi$ minimizing $\bar{\epsilon}$ (2.17) for every given query $\mathbf{q}$, we could alternatively look for a query-independent mapping function $\mathbf{Q} : U_V \rightarrow U_Q$ (e.g., Fig. 2.7 (c)(d)) which minimizes $\bar{\epsilon}$ and satisfies a following condition,

$$d_{obj}(\mathbf{q}, \mathbf{r}) = d_\psi(\psi_q(\mathbf{r})) = d_Q(\mathbf{Q}(\mathbf{v})). \tag{2.28}$$

In other words, the problem of designing low complexity d_ψ and finding the optimal ψ_q given $\mathbf{q}$ and R can be replaced by the problem of designing low complexity d_Q and finding the optimal $\mathbf{Q}$ that minimize $\bar{\epsilon}$ given F_V. This is because $\mathbf{Q}$ is query-independent which allows off-line process to find the optimal $\mathbf{Q}$ and also because $\bar{\epsilon}$ of $\psi_q : U \rightarrow U_\psi$ is identical to $\bar{\epsilon}$ of $\mathbf{Q} : U_V \rightarrow U_Q$.

To minimize overhead cost of converting optimal $\mathbf{Q}$ to determine optimal ψ_q prior to each querying process, we use the same metric for d_Q and d_ψ ($d_Q = d_\psi$) and design $\mathbf{Q}$ to be analogous to ψ_q (e.g., Fig. 2.9 (c)) as follow:

$$\mathbf{Q}(\mathbf{v}) = (Q_1(v_1), Q_2(v_2), \cdots, Q_D(v_D)), \quad \mathbf{v} \in U_V \tag{2.29}$$

where each Q_j is similarly a non-uniform scalar quantization function which is applied independently on j-th dimension v_j of U_V space. Thus, similarly to ψ_q, $\mathbf{Q}$ partitions the viewpoint space U_V into a set of hyper-rectangular cells U_Q (e.g., Fig. 2.7 (d)), where each cell is represented with a vector of mapping symbols $\mathbf{Q}(\mathbf{v})$ and we denote it by $\mathbf{z} := \mathbf{Q}(\mathbf{v}) \in U_Q$. Once the optimal $\mathbf{Q}$ that minimizes $\bar{\epsilon}$ and satisfies above design conditions is obtained 'off-line', given a query $\mathbf{q}$ prior to each querying process, optimal ψ_q can be obtained by the following simple equation:

$$\psi_{qj}(r_j) = Q_j(v_j) = Q_j(d_j(q_j, r_j)), \quad \forall j \tag{2.30}$$

For example, let a scalar quantizer Q_j divides a j-th dimension into a set of intervals $S = \{s_i; s_i = [\theta_i, \theta_{i+1})\}$ covering all possible values with a set of thresholds $\{\theta_i\}_i$ and assigns a mapping symbol m_i to each interval s_i.

$$Q_j(v_j) = \sum_i m_i 1_{s_i}(v_j) \tag{2.31}$$

If d is ℓ_p norm for instance, a scalar quantizer ψ_{qj} is determined according to (2.30) such that

$$\psi_{qj}(r_j) = \sum_i m_i 1_{\sigma_i}(r_j) \tag{2.32}$$

with a set of ψ_{qj} quantization thresholds $\{q_j \pm \sqrt[p]{\theta_i}\}_i$ and a set of its corresponding intervals

$$\Sigma = \{\sigma_i; \sigma_i = [q_j + \sqrt[p]{\theta_i}, q_j + \sqrt[p]{\theta_{i+1}}) \cup [q_j - \sqrt[p]{\theta_{i+1}}, q_j - \sqrt[p]{\theta_i})\}$$

In other words, first find a set of thresholds $\{\theta_i\}_i$ for the optimal Q_j and store $\{\sqrt[p]{\theta_i}\}_i$ off-line, and then for every new given query $\mathbf{q}$, a new set of thresholds $\{q_j \pm \sqrt[p]{\theta_i}\}_i$ for ψ_{qj} needs to be updated on the fly. Note that this computation for updating ψ_{qj} is done only once given a query $\mathbf{q}$ before computing any d_{obj} for all data points to identify $\mathbf{q}$'s NN, which only costs negligible overhead.

2.4.3.5 Q-space vs. target space

Even though query adaptive quantization based NNS metric is our target metric d_{obj} which maps original metric space to target space (U_ψ, d_ψ),

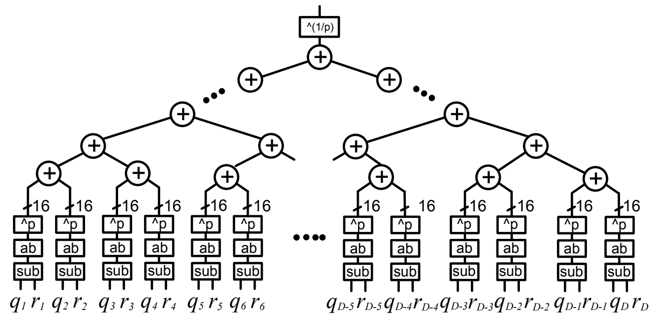
$$d_{obj}(\mathbf{q}, \mathbf{r}) = d_\psi(\psi_q(\mathbf{r}))$$

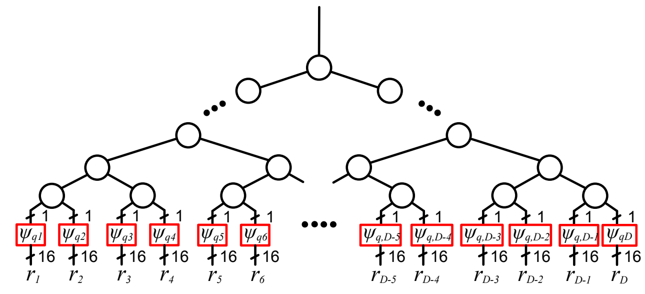
similarly, one can also use query *independent* quantization based NNS metric which maps original metric space to Q-space.

$$d_{obj2}(\mathbf{q}, \mathbf{r}) = d_Q(Q(|\mathbf{q} - \mathbf{r}|))$$

Both have their pros and cons. Since the former metric d_{obj} uses threshold values that change depending on query data, there is more restriction in optimizing hardware circuit complexity. However, the latter metric d_{obj2} use always fixed thresholds, therefore, hardware circuit complexity can be even further simplified thus it is more appropriate with dedicated chip. However absolute difference between query and objects need to be computed a prior.

Fig. 2.9 illustrates the metric computation hardware architecture of Minkowski metric. Fig. 2.9 (a) represents the original Minkowski metric d, Fig. 2.9 (b) represents our target metric d_A where query adaptive quantizer ψ_q is used. Fig. 2.9 (c) shows equivalent metric of Fig. 2.9 (b) but using global quantizer Q_j. Fig. 2.9 (b) and (c) shows that 8 or 16 bit-depth databus interconnect is quantized into 1 bit. Although this is just an example, almost all of our simulations are based on 1 bit quantizer (per dimension) which we find it sufficient.


(a)


(b)

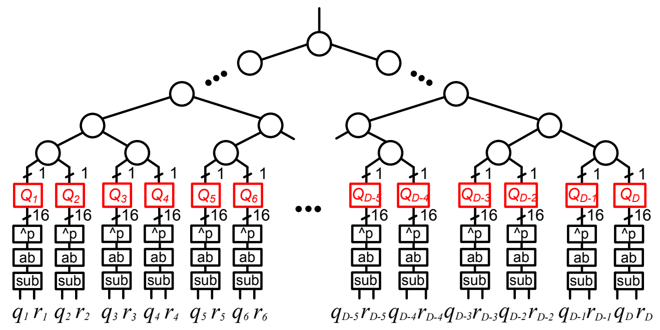

(c)

Figure 2.9: Example illustration of hardware architectures of three metric computations. (a) *Minkowski* metric $d(\mathbf{q}, \mathbf{r}) = (\sum_j |q_j - r_j|^p)^{1/p}$ is shown as an example original metric d. (b) shows proposed *QNNM* metric $d_{obj}(\mathbf{q}, \mathbf{r}) = d_\psi(\psi_q(\mathbf{r}))$ which approximates d. ψ_{qj} is a query-dependent non-uniform scalar quantizer which compresses input to a fewer bits (typically 8 or 16 bits into 1-bit), replacing $|q_j - r_j|^p$ computation in (a). Blank circle represents an operator determined by d_ψ, e.g., it can be an adder if $d_\psi(\mathbf{x}) = \sum x_j$, comparator if $d_\psi(\mathbf{x}) = max(x_j)$, or logical operator *OR* if $d_\psi(\mathbf{x}) = \vee x_j$. (c) is the equivalent metric of d_{obj} of (b), represented with query-independent quantizer Q_j. Q_j minimizing $\bar{\epsilon}$ (2.17) is found via off-line optimization and used to determine ψ_{qj} which equivalently minimizes average NNS error $\bar{\epsilon}$.

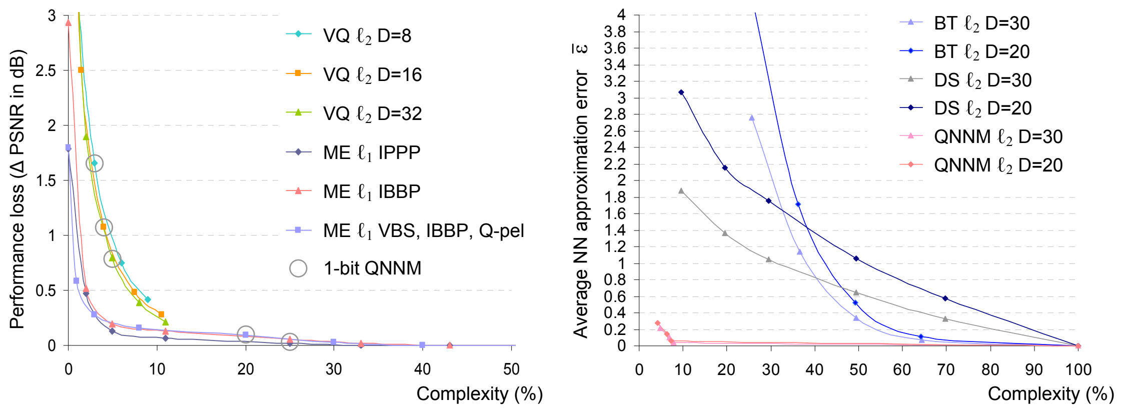


Figure 2.10: Trade-off between the complexity ($cost$) and NNS accuracy degradation ($\bar{\epsilon}$), e.g., the coarser quantization causes higher NNS distortion $\bar{\epsilon}$ with lower complexity $cost$ while the finer quantization leads to lower $\bar{\epsilon}$ at the expense of higher $cost$. Different applications (i.e., different F_V) result in different trade-off curves. The right-hand side of the curve represents the region of inefficient choices of Q and d_Q while the left side represents infeasible choices of Q and d_Q design. Our goal is to design Q and d_Q given F_V and $\bar{\epsilon}_{tol}$ such that its resulting complexity and $\bar{\epsilon}$ pair correspond to the red points on the curves, achieving the lowest complexity with $\bar{\epsilon} \leq \bar{\epsilon}_{tol}$. (Left) *QNNM* performance: the trade-off curve between error in accuracy ($\Delta PSNR$ in dB) and complexity. *QNNM* is applied to the vector quantization process for image coding & motion estimation process for video coding applications. (Right) With the data set $R \sim \mathcal{N}_{\mathcal{D}}(0, \Sigma)$ with $\rho_{ij} = 0.5$, $\sigma^2 = 100$ for all i, j, this compares the performance of three approximate NNS methods: bit truncation (BT), dimension subsample (DS), and *QNNM*.

In Section 2.5, we develop and describe the optimal quantizer design which minimizes $\bar{\epsilon}$ given a fixed complexity budget based on all design conditions described above.

2.5 Finding the Optimal Q

The complexity reduction comes at the expense of some performance loss due to the quantization process. As expected, there is a trade-off between complexity and performance such that coarser quantization will lead to further complexity reductions, while increasing degradation in search performance. To maximize the performance for a fixed number of quantization levels or steps, it is critical to find the optimal quantizer.

As shown in Section 2.4, the problem of finding the optimal target metric space (U_ψ, d_ψ) for the proposed d_{obj} can be replaced by the problem of finding a quantized viewpoint space (U_Q, d_Q) which minimizes $\bar{\epsilon}$ and complexity of its mapping function Q and d_Q metric. The design of (U_Q, d_Q) consists of determining d_Q metric and a quantization based mapping function Q (2.29). Q is a set of scalar quantizers $\{Q_j\}_j$, for each Q_j of which includes three sets of parameters; the number of quantization levels b_j, a set of quantization thresholds $\{\theta_{ji}\}_i$, and a set of mapping symbols $\{m_{ji}\}_i$. $\{b_j\}$ and $\{\theta_{ji}\}$ determines how to partition a viewpoint space (U_V, d_V) into a set of hyper-rectangulars while $\{m_{ji}\}_i$ and d_Q determines the ranking of each hyper-rectangular in terms of the probability of having NN.

The whole design of d_Q, $\{b_j\}$, $\{\theta_{ji}\}$, and $\{m_{ji}\}$ should be chosen so as to maximize NNS performance in both search accuracy and computational complexity. Finding such optimal *Q-space* design depends highly on the input data distribution F_V (2.27) and the degree of NNS error tolerance $\bar{\epsilon}_{tol}$ of a given application. For example, in some applications (e.g., motion estimation for video coding in Section 2.7.1), coarsest 1-bit quantizer Q_j and a simple d_Q metric with logic operators OR ($d_Q(z) = \vee z_j$) still results in negligible performance degradation while some other applications (e.g., vector quantization for image coding in Section 2.7.2) could require 2-bit quantizer Q_j with $d_Q(\mathbf{z}) = \sum z_j$ to achieve similar performance.

Note that since NNS error tolerance level $\bar{\epsilon}_{tol}$ is application-specific, those applications with very small $\bar{\epsilon}_{tol}$ may not benefit much from our proposed $QNNM$ technique. A user can estimate the performance of $QNNM$ when used within a given application of interest to decide whether to use $QNNM$, without necessarily having to find the optimal Q and d_Q functions first (refer to Section 2.6.2 on how). Once a user decide to use $QNNM$, one could proceed to find the optimal Q function.

In this section, we present an optimization algorithm that finds the optimal mapping function Q^* and metric d_Q^* which minimize both average NNS accuracy degradation $\bar{\epsilon}$ and computational complexity $cost$ of $d_{obj} = d_Q(Q(v))$(2.28).

$$\left(Q^*, d_Q^*\right) = \underset{\left(Q, d_Q\right)}{\operatorname{argmin}} \; [\bar{\epsilon}, cost]^T \tag{2.33}$$

Simultaneously optimizing two conflicting objective functions, $f_{obj1} = \bar{\epsilon}$ and $f_{obj2} = cost$ produces a set of *Pareto* optimal solutions as shown in Fig. 2.10 trade-off curves between $\bar{\epsilon}$ and $cost$. Our design goal is to find a *Pareto* optimal solution whose resulting $\bar{\epsilon}$ is within an acceptable range ($\bar{\epsilon} \leq \bar{\epsilon}_{tol}$) while $cost$ is minimized.

Optimization process in general consists of two phases: the search process (i.e., generating candidate solutions) and the evaluation process (evaluating solutions, i.e., f_{obj} computation). Note that in the context of this optimization, we refer to the *'search'* for the optimal set of quantizer design parameters, which should not be confused with the search performed in NNS itself.

This problem (2.33) is a stochastic optimization (SO) problem [1] [49] since both input and output of a system/objective function ($f_{obj1} = \bar{\epsilon}$ in particular) involve stochastic behavior and it aims to achieve optimality on average. In other words, evaluating f_{obj1} can be only estimated, typically through Monte-Carlo simulation approach (e.g., training data samples are simulated to evaluate the average NNS

[1]This contrasts with the conventional deterministic optimization where the values of the objective function are assumed to be exact. While SO algorithm is also often referred to as randomized search methods which incorporate randomness in search algorithm, SO algorithm in this thesis refers to optimization methods for stochastic objective functions.

performance $f_{obj1} = \bar{\epsilon}$). Thus, unlike conventional deterministic optimization problems, reliable evaluation of stochastic f_{obj} alone imposes serious computational burden. Moreover, most of useful assumptions/properties (smoothness, continuity, differentiability, linearity, convexity, unimodality etc.) for designing efficient search process tend to be unavailable, which consequently leads to extra complication in designing the search process of finding a global minimum. Furthermore, this optimization problem often involves searching high dimensional solution space/search space [2]. These three characteristics (expensive evaluation process, difficult searching process, and large search space) make this stochastic optimization problem extremely computationally expensive.

Therefore our goal is to design a highly specialized optimization algorithm for our problem to minimize such computational cost as much as possible.

2.5.1 Proposed Stochastic Optimization Algorithm to Find the Optimal Q

Proposed algorithm reduces complexity by formulating f_{obj1} such that a large portion of f_{obj1} computations can be shared and computed only once as a preprocessing step for a certain set of (quantizer) solution points, instead of computing f_{obj1} for each solution point independently. This leads to the total optimization complexity to change from $O(TN_s)$ to $O(T + c_1 + c_2N_s)$, where T is the size of training data which needs to be sufficiently large, N_S is the total number of candidate solutions evaluated during the search process. c_1 and c_2 are preprocessing cost and f_{obj1} evaluation cost, respectively. This requires a joint design of the search and evaluation processes.

[2]Note that search space of (2.33) should not be confused with metric space U or U_ψ(2.14) of NNS problem. Each dimension of the search space of (2.33) represents θ_{ji}, m_{ji}, or d_Q.

2.5.1.1 Modeling input data set distribution

We first need to model input data set of interest R and its corresponding viewpoint space distribution f_V. If we denote the distribution of data set R by f_R, f_V can be derived from autocorrelation of f_R or cross-correlation of f_R and f_q if the distribution of queries f_q is different from data set distribution f_R.

For example, if dimensional distance $d_j(q_j, r_j)$ of original metric $d(\mathbf{q}, \mathbf{r})$ as in (2.16) is the form of $w_j\,|q_j - r_j|^p$, the probability density function (pdf) of $|\mathbf{q} - \mathbf{r}|$ is (denoted by $f_{|\mathbf{q}-\mathbf{r}|}$) as:

$$f_{|\mathbf{q}-\mathbf{r}|}(\mathbf{a}) = \sum_{|\mathbf{x}|=\mathbf{a}} (f_R \star f_R)(\mathbf{x}) \quad \mathbf{q}, \mathbf{r} \sim f_R,\ \mathbf{a} \geq 0 \tag{2.34}$$

$$F_{|\mathbf{q}-\mathbf{r}|}(\mathbf{a}) = \int_{-|\mathbf{a}|}^{|\mathbf{a}|} (f_R \star f_R) d\mathbf{x} \quad \mathbf{q}, \mathbf{r} \sim f_R,\ \mathbf{a} \geq 0 \tag{2.35}$$

$$f_{|\mathbf{q}-\mathbf{r}|}(\mathbf{a}) = \sum_{|\mathbf{x}|=\mathbf{a}} (f_q \star f_R)(\mathbf{x}) \quad \mathbf{q} \sim f_q,\ \mathbf{r} \sim f_R,\ \mathbf{a} \geq 0 \tag{2.36}$$

$$F_{|\mathbf{q}-\mathbf{r}|}(\mathbf{a}) = \int_{-|\mathbf{a}|}^{|\mathbf{a}|} (f_q \star f_R) d\mathbf{x} \quad \mathbf{q} \sim f_q,\ \mathbf{r} \sim f_R,\ \mathbf{a} \geq 0 \tag{2.37}$$

where $\star$ is cross-correlation operator defined as

$$(f \star g)(\mathbf{a}) = \int_{-\infty}^{\infty} f^*(\mathbf{x}) g(\mathbf{a} + \mathbf{x}) d\mathbf{x} \tag{2.38}$$

Then f_V can be represented as

$$f_V(\mathbf{a}) = f_{|\mathbf{q}-\mathbf{r}|}(g^{-1}(\mathbf{a})) \tag{2.39}$$

$$F_V(\mathbf{a}) = F_{|\mathbf{q}-\mathbf{r}|}(g^{-1}(\mathbf{a})) \tag{2.40}$$

where

$$g(\mathbf{x}) = (w_1 x_1^p,\ w_2 x_2^p, \cdots,\ w_D x_D^p)$$

For example, if we model data set R with D-variate normal distribution: $\mathbf{r}, \mathbf{q} \sim \mathcal{N}_{\mathcal{D}}\ (\mu,\ \Sigma)$ with mean vector μ and covariance matrix Σ, then $(\mathbf{q}-\mathbf{r}) \sim \mathcal{N}_{\mathcal{D}}\ (\mathbf{0},\ 2\Sigma)$. If $d = \ell_1$ norm,

$$f_V(\mathbf{a}) = \frac{2}{(2\pi)^{D/2}\,|\Sigma|^{1/2}}\, exp\left(-\frac{\mathbf{a}'\Sigma^{-1}\mathbf{a}}{2}\right) \quad \mathbf{a} \geq 0 \tag{2.41}$$

If $d = \ell_2$ norm,

$$f_V(\mathbf{a}) = \frac{2}{(2\pi)^{D/2}\,|\Sigma|^{1/2}}\, exp\left(-\frac{\mathbf{a}'\Sigma^{-1}\mathbf{1}}{2}\right) \quad \mathbf{a} \geq 0 \tag{2.42}$$

2.5.1.2 Objective function formulation

Given F_V input distribution, our objective function $f_{obj1} = \bar{\epsilon}$ can be formulated using F_V. $f_{obj1} = \bar{\epsilon}$ is an NSS error measure as defined in (2.17), consists of two terms, $d(\mathbf{q}, \mathbf{r}^*(\mathbf{q}))$ and $d(\mathbf{q}, \mathbf{r}^*_\psi(\mathbf{q}))$. Finding a NNS algorithm which minimizes $\bar{\epsilon}$ is the same as finding one that minimizes $d(\mathbf{q}, \mathbf{r}^*_\psi(\mathbf{q}))$ since $E[d(\mathbf{q}, \mathbf{r}^*(\mathbf{q}))]$ term is constant given F_V or dataset R while $E[d(\mathbf{q}, \mathbf{r}^*_\psi(\mathbf{q}))]$ changes with Q parameters. Therefore, f_{obj1} can be reduced to:

$$f_{obj1} = E[d(\mathbf{q}, \mathbf{r}^*_\psi(\mathbf{q}))] = \sum_a \mu_Q(a)\, f_Q^{min}(a) \tag{2.43}$$

where f_Q^{min} is the pdf of $F_Q^{min}(a)$, and

$$F_Q^{min}(a) = Pr(d_{obj}(\mathbf{q}, \mathbf{r}^*_\psi) \leq a), \tag{2.44}$$

$$\mu_Q(a) = E(d(\mathbf{q}, \mathbf{r}) | d_{obj}(\mathbf{q}, \mathbf{r}) = a, \forall \mathbf{q}, \mathbf{r} \in U). \tag{2.45}$$

To compute μ_Q and F_Q^{min}, we first assign three parameters to each cell $c_\mathbf{z}$ of the set of hyper-rectangular cells defined by Q (Fig. 2.7 lower right): (i) probability mass $p_\mathbf{z}$, (ii) non-normalized centroid $u_\mathbf{z}$, and (iii) distance $d_\mathbf{z} = \sum z_j$.

$$p_\mathbf{z} = \int_{c_\mathbf{z}} f_V(\mathbf{v}) d\mathbf{v} \tag{2.46}$$

$$u_\mathbf{z} = \int_{c_\mathbf{z}} < \mathbf{v}, \mathbf{1} > f_V(\mathbf{v}) d\mathbf{v} \tag{2.47}$$

Then F_Q^{min} and $\mu_Q(a)$ are computed as:

$$F_Q(a) = \sum_{d_\mathbf{z} \leq a} p_\mathbf{z} \tag{2.48}$$

$$F_Q^{min}(a) = 1 - (1 - F_Q(a))^N \tag{2.49}$$

$$\mu_Q(a) = \frac{\sum_{d_\mathbf{z}=a} u_\mathbf{z}}{\sum_{d_\mathbf{z}=a} p_\mathbf{z}} \tag{2.50}$$

2.5.1.3 Preprocessing based objective function evaluation

Computing f_{obj1} is simple once $p_\mathbf{z}$, $u_\mathbf{z}$ are known for all cells $c_\mathbf{z}$, but obtaining $p_\mathbf{z}$, $u_\mathbf{z}$ in the first place can be complex. However, if the following two data sets F_V and H_V are available or computed in a pre-processing stage:

$$F_V(\mathbf{x}) = Pr(\mathbf{v} \leq \mathbf{x}) \tag{2.51}$$

$$H_V(\mathbf{x}) = \sum_{\mathbf{v} \leq \mathbf{x}} < \mathbf{v}, \mathbf{1} > \tag{2.52}$$

then $P_\mathbf{z} = \sum_{\mathbf{z}' \leq \mathbf{z}} p_{\mathbf{z}'}$ and $U_\mathbf{z} = \sum_{\mathbf{z}' \leq \mathbf{z}} u_{\mathbf{z}'}$ can be easily computed for each cell $c_\mathbf{z}$, so

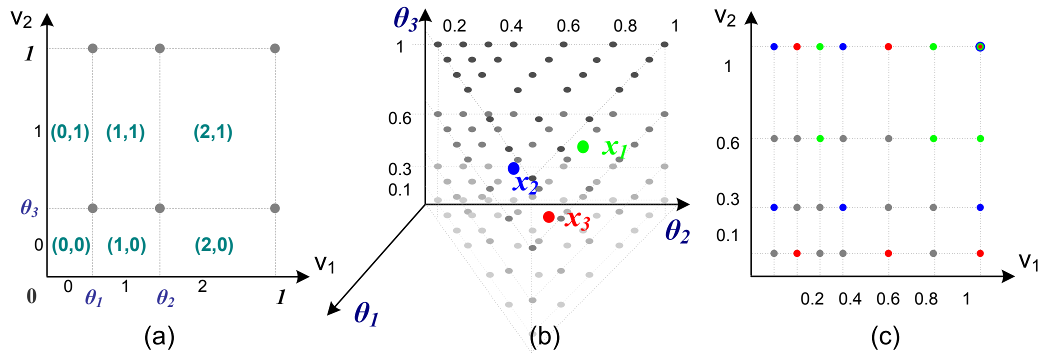


Figure 2.11: (a) A simple example of 2D viewpoint space partitioned by $Q = \{\theta_1, \theta_2, \theta_3\}$. P_z, U_z values at all six gray points need to be retrieved from F_V, H_V to compute f_{obj}. (b) 3D search space of Q. x_1, x_2, x_3 are 3 arbitrarily chosen candidate solutions for Q. (c) shows the preprocessed structures (F_V, H_V having P_z, U_z values at all points in (c)) to compute f_{obj} for x_1, x_2, x_3. However, f_{obj1} for all gray points in (b) can be computed with the same F_V, H_V in (c).

that all necessary $p_{\mathbf{z}}$, $u_{\mathbf{z}}$ values can be obtained with only $c_2 = O(DN_C)$ cost, where D is dimension and N_C is total number of cells generated by Q. $N_C = \prod_j (b_j + 1)$, where b_j denotes the number of thresholds assigned by Q on j-dimension of U_V.

However, the computational (c_1) and storage complexity of F_V and H_V increase exponentially (e.g., $O(DW^D)$ assuming all dimensions are represented with the same resolution W). To reduce such complexity, while it is very important to minimize the dimensionality D if possible [3], it is also important to note that only a small fraction of F_V and H_V data relating to the candidate solutions on the search path is used during the optimization process.

We next propose a search algorithm that maximally reuses F_V and H_V data and show how F_V and H_V can be updated in conjunction with the search process in order to reduce overall storage and computation.

[3] D is reducible depending on the input distribution F_V if certain dimensions are independent or interchangeable/commutative. In fact this is usually the case for real-world applications (e.g., for video coding, all pixels tend to be heavily correlated yet interchangeable statistical characteristics thus common 16×16 processing unit image block (D=256) can be reduced to D=1).

2.5.1.4 Preprocessing and search process co-design

Given k arbitrary solution points on the search space, preprocessing cost S_k to build F_V and H_V containing only necessary data to compute f_{obj1} of those k points is the same as that for computing f_{obj1} of K different solution points G which form a *grid*, where:

$$K = \prod_j \begin{pmatrix} (k+1)b_j \\ b_j \end{pmatrix} \qquad S_k = \prod_j (kb_j + 1) \tag{2.53}$$

In other words, if every adjacent pair of solution points in G can be connected by line of equal size Δ, all such regularly spaced lines form a D_S-dimensional grid in D_S-dimensional search space. If so, evaluation of solution points in G can maximally reuse data from F_V and H_V and thus lead to minimal preprocessing cost in both space and time complexity.

Fig. 2.11 provides a simple example which illustrates that the minimum required preprocessing cost to compute f_{obj1} for a set of three *arbitrary* solution points (e.g., x_1, x_2, x_3 in Fig. 2.11 (b), k=3) is the same as that for a set of 112 solution points (e.g., all gray points in Fig. 2.11 (b), $K = 112$) which forms a *grid* structure.

Based on the above observation and the unimodality of f_{obj1} function over the search space [4], we describe a grid based iterative search algorithm framework with guaranteed convergence to the optimal solution [5].

The basic iteration of this algorithm consists of (i) generating a grid G_i which equivalently indicates a set of solution points which correspond to all grid points, (ii) building minimum required preprocessed structures F_{Vi} and H_{Vi} for computing

[4]We represent each quantization parameter not with the actual threshold value θ but with the marginal cumulative probability $F_V(\theta)$, such that the search space becomes $[0, 1]^D$.This is not only for ease of f_{obj} computation but also helps increasing slope, reducing neutrality, ruggedness, or discontinuity of f_{obj} function, leading to higher search speed-up and making f_{obj} unimodal. This also provides further indication regarding to the sensitivity to performance.

[5]its convergence result is similar to those in [51], [31]

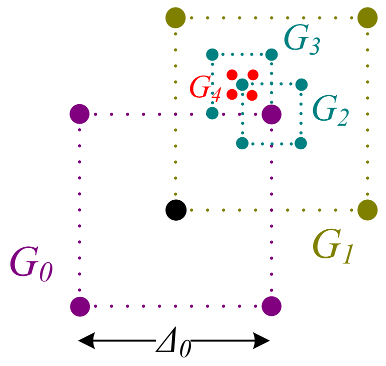


Figure 2.12: An example illustration of grid-based optimization algorithm with ω=*2*, γ=*3* in 2*D* search space. It searches for the optimal Q^* (quantization parameters) for *QNNM* (2.15). It shows the iterations of either moving or $1/\gamma$-scaling a grid G of size $\omega \times \omega$. Note that this should not be confused with ME search or NNS search algorithms.

f_{obj} of all grid points on G_i, (iii) computing a set of f_{obj1} and finding its minimizer Q_i^* of G_i, and (iv) generating a next grid G_{i+1} by either moving or scaling G_i based on Q_i^* information.

We model a grid G on the search space with its center/location C, grid spacing Δ, and size parameter ω, assuming it has equal spacing and size for all dimensions, such that the total number of solution points in G is to be ω^{D_S}.

With initialization of grid-size parameter ω, grid scaling rate γ, tolerance for convergence $\Delta_{tol} > 0$, grid-spacing parameter Δ_0, and initial grid G_0, for each iteration $i = 0, 1, ..$

1. **Preprocess**: construct F_{Vi} and H_{Vi} to evaluate G_i
2. **Search**: seek a minimizer Q_i^* among the points in G_i
3. **Update**: generate a new grid G_{i+1} based on Q_i^*
   - *Move the center of grid*: $C_{i+1} = Q_i^*$
   - *Grid space update*
     - ***Moving grid***: if Q_i^* is on the boundary of grid G_i:

$$\Delta_{i+1} = \Delta_i$$

- ***Scaling grid***: if Q_i^* is not on the boundary of grid G_i:

$$\Delta_{i+1} = \Delta_i/\gamma$$

- ○ *Terminate*: if $\Delta_{i+1} < \Delta_{tol}$
- ○ *Generate* G_{i+1}: with parameters ω, Δ_{i+1}, and C_{i+1}

Given this algorithm modeling, our goal is to find two integer parameter values, w and γ, minimizing overall computational complexity.

Overall optimization complexity can be quantified as [6]:

$$O(T + Lc_1 + Lc_2 N_s), \tag{2.54}$$

$$c_1 = O(D\omega^B), \tag{2.55}$$

$$c_2 = O(DS_1), \tag{2.56}$$

where c_1 is both time and space complexity of phase-1 search. L denotes the total number of iterations. Note that c_2 is fixed regardless of ω and γ. N_s depends on phase-2 grid search algorithm which is described below, but roughly varies from $O(\omega D)$ to $O(\omega c^D)$.

If we assume to continue iteration until it gets as fine as resolution W, total iteration number is $L \approx \frac{\gamma}{\omega} \log_\gamma \frac{W}{w}$. Therefore, $\gamma \geq 1$ minimizing $\gamma \log_\gamma W$ and minimum possible integer $\omega \geq 2$ minimizes overall complexity in both time and space: That is, $\gamma = 3$ and $\omega = 2$. (Fig. 2.12)

[6]Overall complexity can be further reduced from $O(L(T + c_1 + c_2 N_s))$ to $O(T + Lc_1 + Lc_2 N_s)$ by splitting and deleting portions of training data set at each iteration such that only relevant data is examined for each update.

Phase-2 Grid Search Algorithm

Given grid points with general $k > 1$ and Δ, the general grid search algorithm can be represented in this form,

$$Q_{n+1} = \Pi_\Theta(Q_n - a_n \hat{\nabla}_n(Q_n))$$

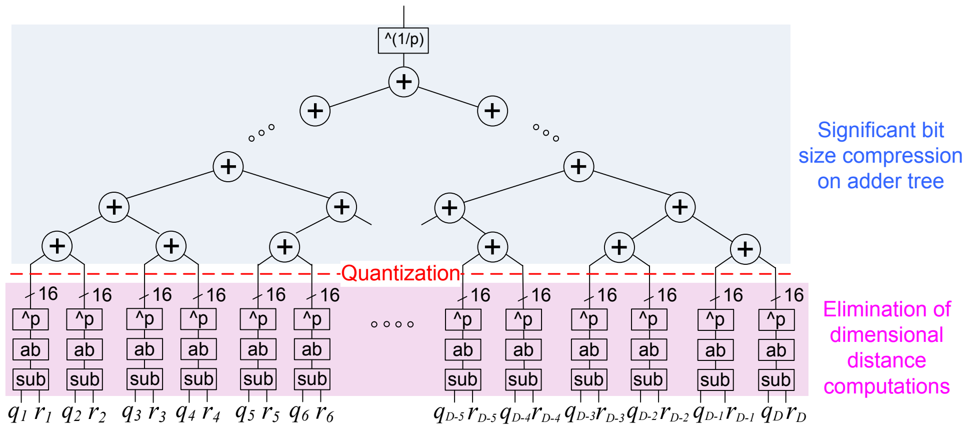
where step size a_n, Π_Θ a projection of points outside Θ back into Θ, search direction $\hat{\nabla}_n$, and iteration n. Note that in determining a_n and $\hat{\nabla}_n$ for this problem, typical stochastic or gradient approximation methods are not useful in general. Due to relatively large Δ, it is unreliable to approximate gradient and often k is small enough to perform line search. Thus $a_n = \Delta$. In determining $\hat{\nabla}_n$, especially as dimensionality increase, stochastic choice of $\hat{\nabla}_n$ tend to be close to orthogonal to true gradient direction which leads to being slow in convergence. Furthermore, randomized choice does not use given information most efficiently (e.g., cycling, visiting similar wrong solutions). Therefore efficient direct search is more suitable.

2.6 Complexity-Performance Analysis

2.6.1 Complexity Analysis

Comparing two metric computations: the original/benchmark Fig. 2.9 (a) and the proposed Fig. 2.9 (c), the proposed approach leads to complexity reductions in i) the summation process after quantization (upper part of the dashed quantization line in Fig. 2.13) and ii) a set of dimension-distance computation processes prior to quantization (lower part of the line in Fig. 2.13).

Fig. 2.14 and Table. 2.2 provide useful insight to understand the computational complexity of the search at the circuit level [7]. Fig. 2.14 illustrates the structure



^(1/p)
Significant bit size compression on adder tree
Quantization
16
^p
ab
sub
Elimination of dimensional distance computations
q_1 r_1 q_2 r_2 q_3 r_3 q_4 r_4 q_5 r_5 q_6 r_6
q_{D-5} r_{D-5} q_{D-4} r_{D-4} q_{D-3} r_{D-3} q_{D-2} r_{D-2} q_{D-1} r_{D-1} q_D r_D

Figure 2.13: The proposed approach leads to complexity reductions in i) the summation process after quantization (significant circuit complexity reduction due to bit size compression), and ii) a set of dimension-distance computation prior to quantization (dimension-distance computation becomes unnecessary).

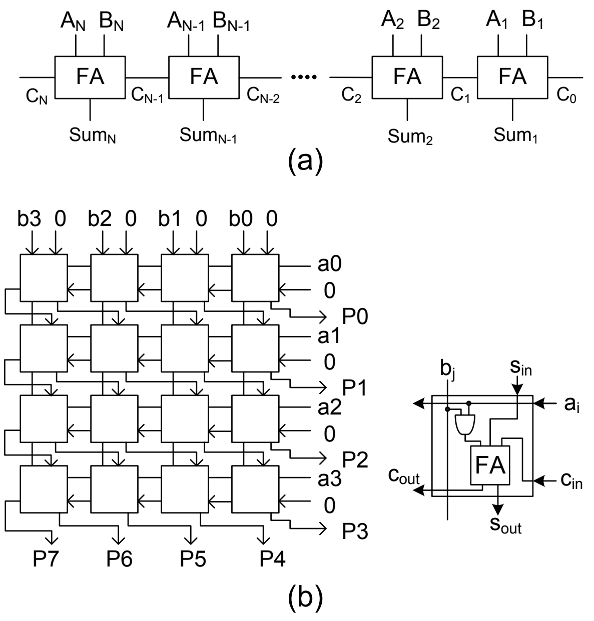


Figure 2.14: Diagrams of arithmetic circuits for (a) N-bit ripple carry adder and (b) 4 bit array multiplier.

Adder Type	# of gates			avg. # of logic transitions/add		
	adder size (bits)			adder size (bits)		
	16	32	64	16	32	64
Ripple Carry	144	288	576	90	182	366
Carry Lookahead	200	401	808	100	202	405
Carry Skip	170	350	695	108	220	437
Carry Select	284	597	1228	161	344	711
Conditional Sum	368	857	1938	218	543	1323

(a)

Multiplier Type	# of gates			avg. # of logic transitions pipelined		
	multiplier size (bits)			multiplier size (bits)		
	8	16	32	8	16	32
Array	528	2336	9792	548	7191	99062
Modified Array	567	2405	9918	583	7348	99102
Wallace	613	2569	10417	573	3874	19548
Dadda	612	2477	10051	557	3389	16638

(b)

Table 2.2: The average number of transitions (a measure of dynamic power consumption) and the number of gates (a measure of circuit size, static power consumption) for various types of adders (a) and multipliers (b) for different input bit sizes. [7]

of arithmetic circuits for a representative N bit-depth adder Fig. 2.14 (a) and 4 bit-depth multiplier Fig. 2.14 (b), the size of which increases significantly with the input bit size. A block FA (full adder) is a single bit adder and a typical N bit adder consists of N full adders linked together by the carry inputs and outputs. Typical N bit multiplier consists of N by N array of cells, each of which consists of full adder and AND gate. Table. 2.2 (b) and (d) essentially demonstrate that the computational complexity, circuit size, static and dynamic power consumption, computation delays of most basic arithmetic elements including adder or multiplier are all directly influenced by, and increase polynomially with, the input bit size.

Therefore quantization applied to dimension-distance terms in each dimension, as shown in Fig. 2.13, leads to significant simplification of the summation process (binary adder tree). Typically 8, 16, or 32 bit-depth input is quantized into either

0,1, or 2 bits depending on the input distribution on each dimension (non-uniform bit allocation over dimensions). For example, for motion estimation in video coding, very coarse quantization (e.g., 1 bit per pixel) has been shown to be sufficient to achieve video coding performance nearly unchanged (average 0.01dB loss. See Section 2.7.1).

Fig. 2.9 (b) and Fig. 2.9 (c) are equivalent metric. As our target metric implementation structure Fig. 2.9 (b) shows, entire set of dimension-distance computations $\{d_j(q_j, r_j)\}_{j=1}^{D}$ and its corresponding circuits can be eliminated.

Comparing the original/benchmark metric computation

$$d(\mathbf{q}, \mathbf{r}) = \sum_{j=1}^{D} d_j(q_j, r_j)$$

and our proposed metric computation architecture

$$d(\mathbf{q}, \mathbf{r}) = \sum_{j=1}^{D} \psi_{qj}(r_j),$$

complexity savings depend on how complex the original metric is and how much complexity budget the user constrains to design the optimal ψ_{qj} (e.g., total number of quantization bins).

Fig. 2.15 illustrates the complexity increase as a function of the input bit size, dimensionality, and order p of metric (if p-norm metric is considered) for both conventional and proposed metric computations. We measure complexity in units of number of full-adder operations (basic building blocks of arithmetic logic circuits), under the assumption that n-bit addition, subtraction, and absolute value operations have the same complexity and that a square operation has equivalent

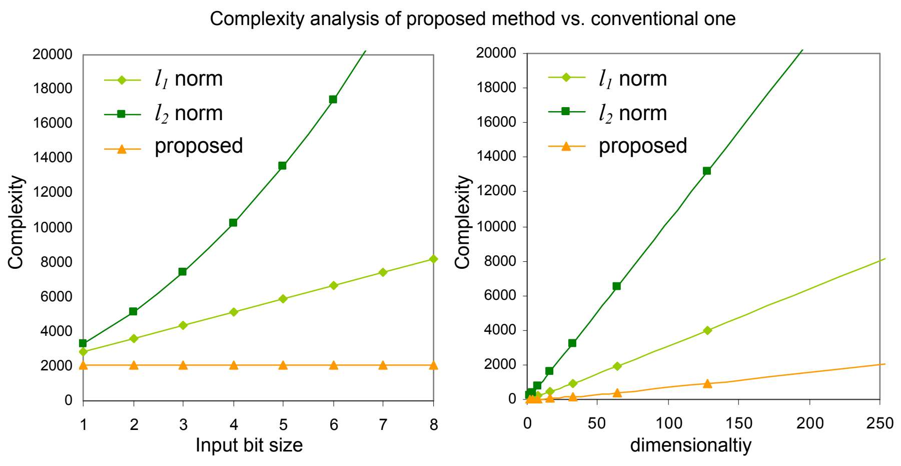


Figure 2.15: Complexity behavior comparison of conventional l_1 and l_2 norm metric computation vs. proposed distance quantization based l_p norm metric computation with respect to the input bit size and dimensionality.

complexity to that of an n^2-bit addition. For motion estimation example, the dimensionality represents the number of pixels per matching block while input bit size represents pixel bit-depth. Note that the complexity of the proposed method remains constant over different input bit sizes and over different original metrics, while it slowly increases with dimensionality as compared to the conventional metric computation.

This approach obviously imposes extra complexity for quantization process. However, this quantization implementation can be integrated with the following adder block such that its overhead cost is kept negligible as compared to the complexity reduction achieved elsewhere.

2.6.2 Performance Analysis

QNNM performance is optimized and evaluated not based on worst-case error measure but on average-case error measure. Therefore, its performance is input data dependent and application specific. In this section, we provide an analysis and identify the statistical characteristics of promising application, as well as simple technique to estimate QNNM performance for any specific application of interest without having to perform any optimization process.

Note that even if NN approximation error $\bar{\epsilon}$ is the same, its system level performance as well as application specific error tolerance level $\bar{\epsilon}_{tol}$ may vary from application to application. Therefore, it is important to estimate how much 'system-level' performance degradation $QNNM$ could introduce as well as complexity saving. To do this, one could first analyze their data set R in terms of its dimensionality, average correlation, original search metric, and average distance to exact NN. Once obtained, refer to Fig. 2.16 and Fig. 2.17 to roughly estimate QNNM error $\bar{\epsilon}$. Then by introducing $\bar{\epsilon}$ error to their NNS system and observing its impact on the system level result, one could use such data to decide whether to adopt QNNM method to one's application of interest or not.

In Fig. 2.16 and Fig. 2.17, data set is modeled by multivariate normal distribution $\mathcal{N}_{\mathcal{D}}(\mu, \pm)$. There are three significant statistical characteristics that influence $QNNM$ performance. These can be shown both analytically and numerically as in Fig. 2.16 and Fig. 2.17.

- Any linear transform of data set would not change QNNM performance in terms of $\bar{\epsilon}$. More specifically any scaling of Σ or any change of μ would not affect $\bar{\epsilon}$ of QNNM.

- QNNM $\bar{\epsilon}$ decays exponentially as dimensionality increases. [7]

[7]Note that dimensionality D here refers to the dimensionality of viewpoint space and it should not be confused with dimensionality of optimization search space D_S. viewpoint space dimensionality is the same as dimensionality of metric space where NN is searched. dimensionality of optimization search space, as we discussed earlier, represents total number of parameters/thresholds used to quantize viewpoint space.

- The more correlated (positive or negative or in any direction) the data set distribution is, the less QNNM $\bar{\epsilon}$ becomes. [8]

Fig. 2.16 and Fig. 2.17 show simulated QNNM performance for four specific data set model settings (multivariate normal distribution with identical marginal distribution and covariance for all dimensions with four different correlation coefficient 0, 0.5, 0.8, 0.95) with various dimensionality with respect to two different original metrics ℓ_1 and ℓ_2. Straight line (no metric) represents average distance per dimension from a query to any randomly chosen object, which can be considered as the worst bound of NNS performance. Blue curves (average distance/D to exact NN) represents the lower bound of NNS performance. Note that average distance to exact NN per dimension is not constant but increases as dimensionality increases. This is due to the weak law of large numbers, where the variance of the distance converges to 0 in probability as dimensionality is sufficiently large. Thus, NNS loses its meaning since all objects converge to the same distance from the query point. While this simulation shows the case with identical correlation for all dimensions and it may not be the case for the actual data set of interest, user can refer to zero correlation (uncorrelation) performance as the worst performance bound of QNNM performance. One can compute average correlation coefficient for all dimensions and can make a rough estimate between curves provided in Fig. 2.16 and Fig. 2.17 and apply estimated $\bar{\epsilon}$ error to the system of interest [9] and determine if the end result is acceptable or not.

[8]Two subspace divided by a hyper-surface of original metric are different from two subspaces divided by a hyperplane of quantization threshold. Common regions of two pairs of split subspaces do not cause error but the remaining regions introduce error. if data distribution is correlated, the ratio of data falling into these error introducing regions is reduced. Therefore, the stronger correlation is in any direction, the smaller $\bar{\epsilon}$ becomes.

[9]instead of searching exact NN, let the system choose the object having distance $=(1+\bar{\epsilon}) * d(NN, \mathbf{q})$

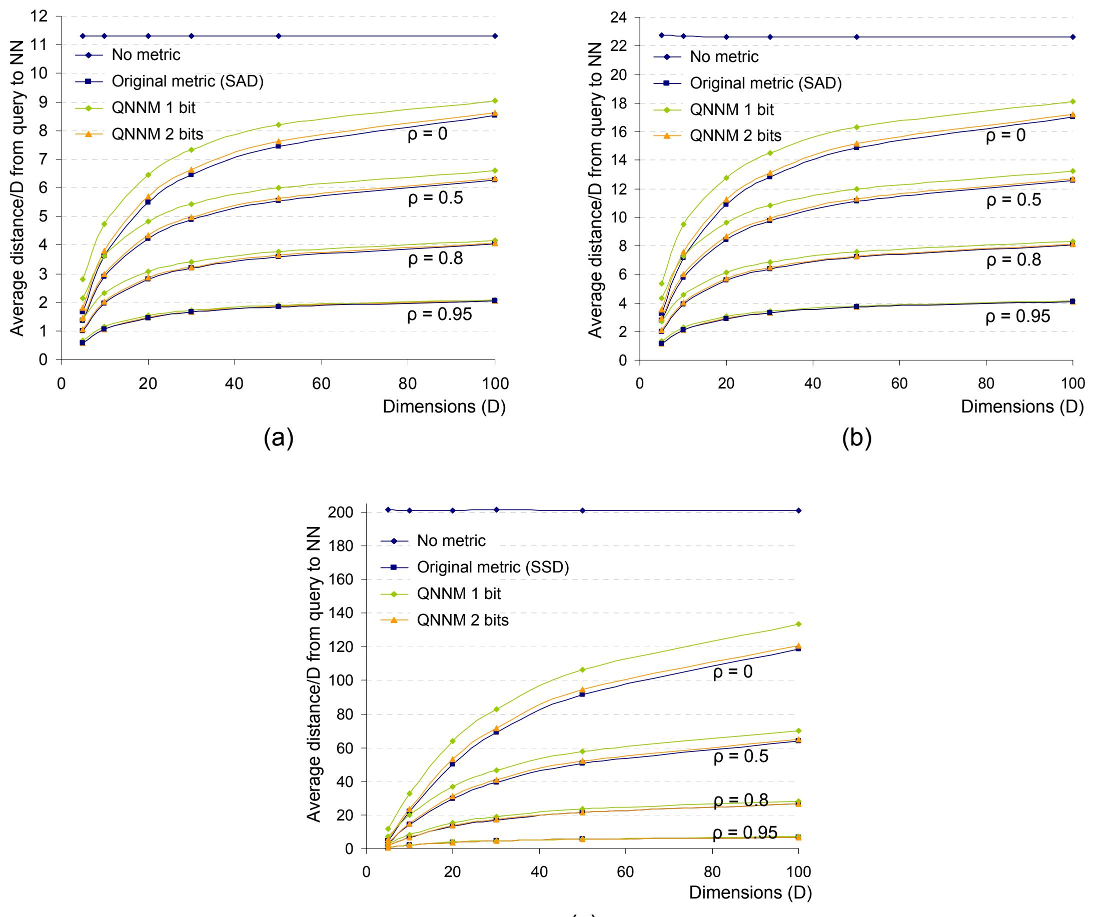


Figure 2.16: QNNM performance when NNS data set R is modeled with multivariate normal distribution $R \sim \mathcal{N}_{\mathcal{D}}(0, \Sigma)$ with different dimensions D and covariance matrices Σ which have identical marginal variances σ_j^2 $\forall j$ and covariance ρ_{ij} $\forall i, j$ for all dimensions. Straight blue line (no metric) represents average distance per dimension from a query to any randomly chosen object, which can be considered as the worst bound of NNS performance. Blue curves (average distance/D to exact NN) represents the lower bound of NNS performance (original metric). (a) ℓ_1 as original metric. $R \sim \mathcal{N}_{\mathcal{D}}(0, \Sigma)$ with σ^2=100 $\forall j$ and ρ_{ij} = 0, 0.5, 0.8, 0.95, $\forall i, j$. (b) The same setting as (a) except σ^2=400 $\forall j$. (c) The same setting as (a) except original metric to be ℓ_2 (Euclidean distance). (a) and (b) show the degree of dispersion of data set of interest does not affect QNNM performance in terms of NN approximation error $\bar{\epsilon}$. However, the stronger the correlation ρ and the higher the dimension, it is more advantageous for QNNM.

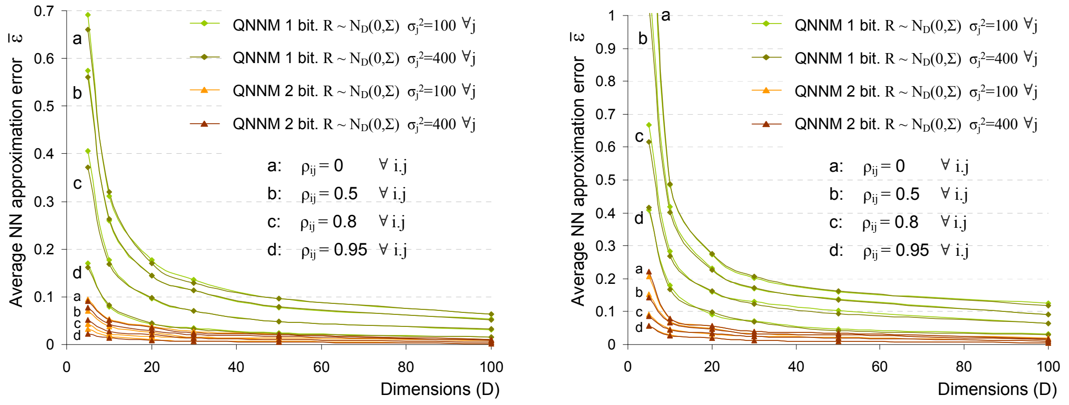


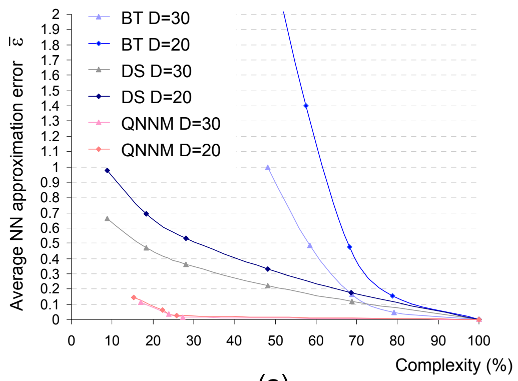
Figure 2.17: Different representation of Fig. 2.16 representing QNNM performance with respect to NNS approximation error $\bar{\epsilon}$ over different dimensionality and covariance. All results are under the same data set model settings as those of Fig. 2.16 with different original/benchmark metric (Left) ℓ_1 and (Right) ℓ_2.

2.7 Example Applications

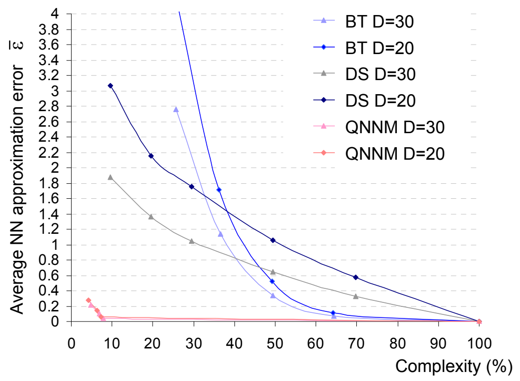
2.7.1 Motion Estimation for Video Compression

In this section, our proposed approach and its analytical study are applied to motion estimation (ME) process used in video coding system as an example application. Experimental results are provided to validate our study and to empirically evaluate the performance of our proposed approach.

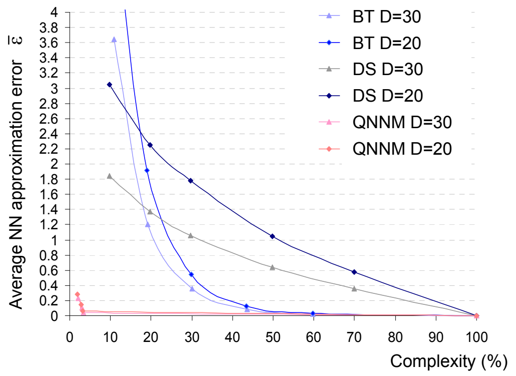
The ME process is one of good example applications for the following reasons: i) its computational burden is very heavy, ii) it is inherently tolerant with search approximation error, iii) dimensions/pixels of data set is highly correlated (Fig. 2.19 (a) image blocks with highly correlated adjacent pixels, and Fig. 2.19 (b) consequently highly correlated viewpoint distance distribution F_V), iv) it holds homogeneity of viewpoint property, which consequently means NN distribution is concentrated in a very narrow range (Fig. 2.19 (c)), v) ME is typically performed on high dimensional data set (from 4×4 to 16×16 block as a vector of dimension



(a)



(b)



(c)

Figure 2.18: Performance comparison of bit truncation methods (BT), dimensional subsampling (DS), and proposed QNNM methods for three different metrics: (a) ℓ_1 norm, (b) ℓ_2 norm, and (c) weighted ℓ_2 norm distance metric. X-axis represents how much metric complexity is reduced in percentage while y-axis represents average performance degradation $\bar{\epsilon}$ in finding NN. The input distribution of data set $R \sim \mathcal{N}_{\mathcal{D}}(0, \Sigma)$ with $\sigma_j^2 = 100$, $\rho_{ij} = 0.5 \;\; \forall i, j$

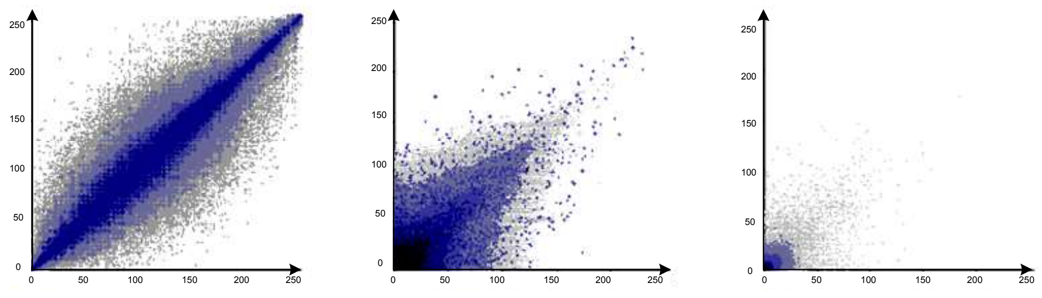


Figure 2.19: 2D statistical distribution of motion estimation data. (a) shows distribution of candidate blocks (in this example, two adjacent pixels) for motion searching, i.e., F in metric space. (b) represents distribution of difference/distance between a query block and candidate blocks, i.e., F_V in viewpoint space. (c) shows distribution of difference/distance between a query block and its best matching block (NN), i.e., F_{NN} in viewpoint space.

16 to 256), and vi) Because F and F_V are identically distributed for each dimension/pixel (Fig. 2.19 (a) (b)), the optimization process of finding the optimal $\mathbf{Q}$ (discussed previously in Section 2.5) can be significantly simplified.

In Fig. 2.20, 9 CIF (352×288) sequences were tested for simulation using a H.264 /MPEG-4 AVC (JM17.1) encoder. Three different ME settings from simple to complex were tested: (i) 16×16 block (D=256) with full-pel accuracy forward prediction only (IPPP), (ii) same as i) but allows bi-directional prediction (IBBP), and (iii) variable block sizes, quarter-pel accuracy, bi-directional prediction setting. Two search metrics: ℓ_1 norm (sum of absolute difference) and proposed 1-bit and 2-bit QNNM were tested. With ℓ_1 norm, full search (with the search window of ±16) is used for motion estimation while with QNNM metric, both full search and a representative fast search algorithm, EPZS were tested. 1-bit QNNM with full search on average results in 0.02dB, 0.06dB, 0.09dB performance loss for ME settings (i),(ii), and (iii), respectively. Similarly 1-bit QNNM with EPZS search results in on average -0.01dB, 0.02dB, 0.02dB performance loss for ME settings (i),(ii), and (iii), respectively. Fig. 2.21 shows for three different ME settings described above

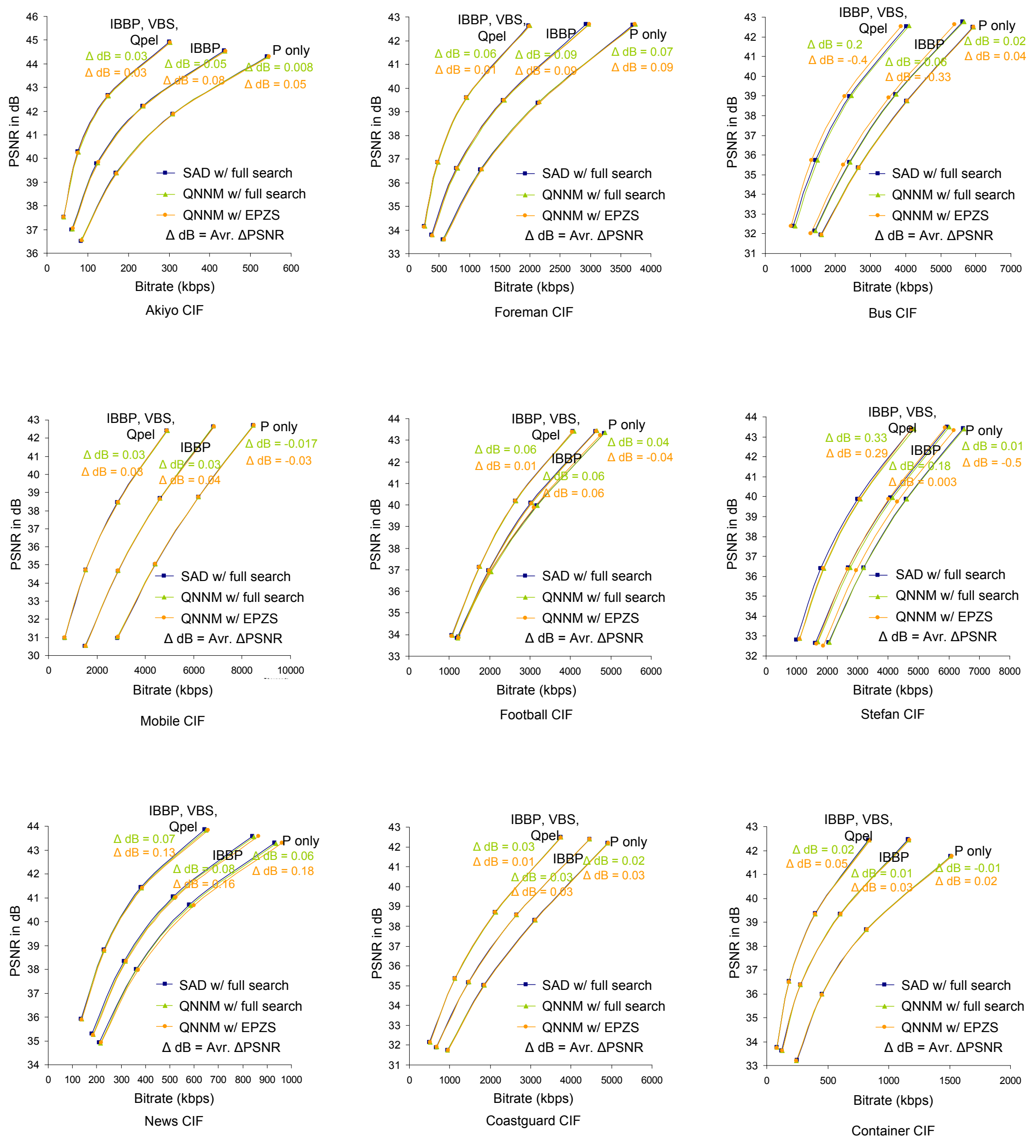


Figure 2.20: Rate-distortion curves for 9 different CIF test sequences. They compare ME performance with original metric (SAD) and with proposed 1-bit QNNM under three different ME settings and two different search algorithms. ΔdB shown represents average performance (PSNR) difference between original metric and 1-bit QNNM.

and for two different search algorithms, how QNNM performance in ME changes with the average number of quantization level per pixel used. The reason QNNM with EPZS in Fig. 2.21 (Right) may perform better than original metric (i.e., negative performance loss when the curve goes below zero) is because original metric SAD itself is an approximation of optimal metric (transform + quantization + entropy coding) for optimal motion vector search.

Both Fig. 2.20 and Fig. 2.21 show QNNM performance with fast search algorithm is as good as or better than QNNM with full search method. Two different search algorithm can be seen as different data set in terms of both distribution and the number of objects, which consequently means different viewpoint distribution as well. However, the reason QNNM works well with very different search algorithms is that distance distributions of NN for both search algorithms are relatively similar. Both Fig. 2.20 and Fig. 2.21 also numerically support our QNNM performance analysis of dimensionality that smaller dimensionality (ME with variable block size use partitioned image block as small as up to $\times 4$ block size) introduces higher performance loss.

Fig. 2.25 illustrates the trade-offs between complexity and performance for proposed and three different representative scenarios. Other sequences tested showed similar results. The proposed approach provides a better trade-off and can also be used together with most of other existing algorithms to further improve the complexity reduction.

Fig. 2.22 (Left) compares our cost function $\bar{\epsilon}$ with the expected performance error collected from numerically simulated experiments for different input distribution settings f_y. As the number of experiments increases, expected error converges to our cost function, confirming the accuracy of our $\bar{\epsilon}$ formulation. Fig. 2.22 (Right)

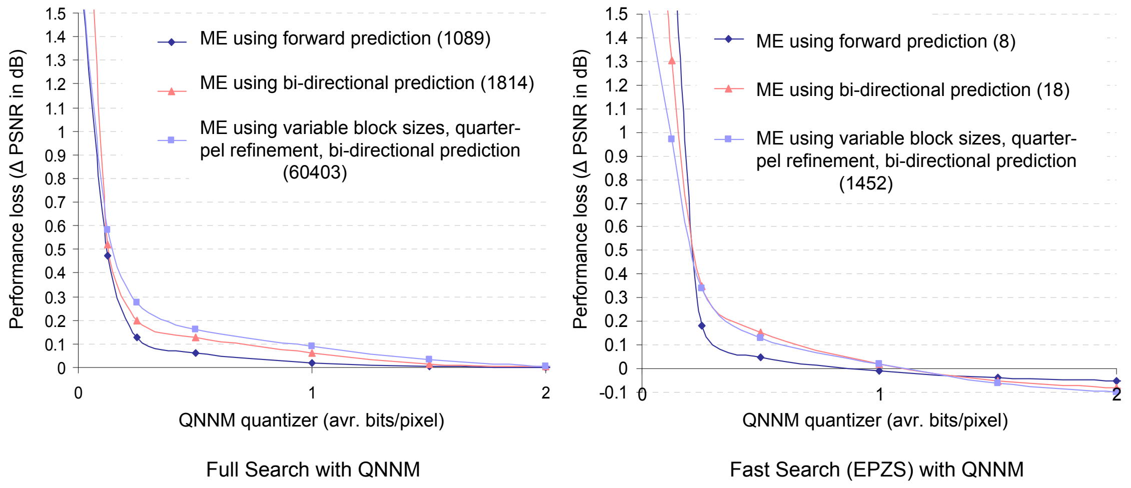


Figure 2.21: Average QNNM performance of 9 different test sequences. QNNM is performed with full search (left) and with a representative fast search method, EPZS (right). Numbers in parentheses shown in graph legend represent the number of metric computation performed per a query macroblock. Three different ME settings from simple to complex were tested: (i) 16×16 block ($D = 256$) with forward prediction only and full-pel, (ii) same as i) but allows bi-directional prediction, and (iii) variable block sizes, quarter-pel accuracy, bi-directional prediction setting. With 1-bit quantizer per dimension/pixel used with full search, on average 0.02dB, 0.06dB, 0.09dB performance loss incurred for ME settings (i), (ii), and (iii), respectively. Similarly 1-bit quantizer used with EPZS results in average -0.01dB, 0.02dB, 0.02dB performance loss for ME settings (i), (ii), and (iii), respectively.

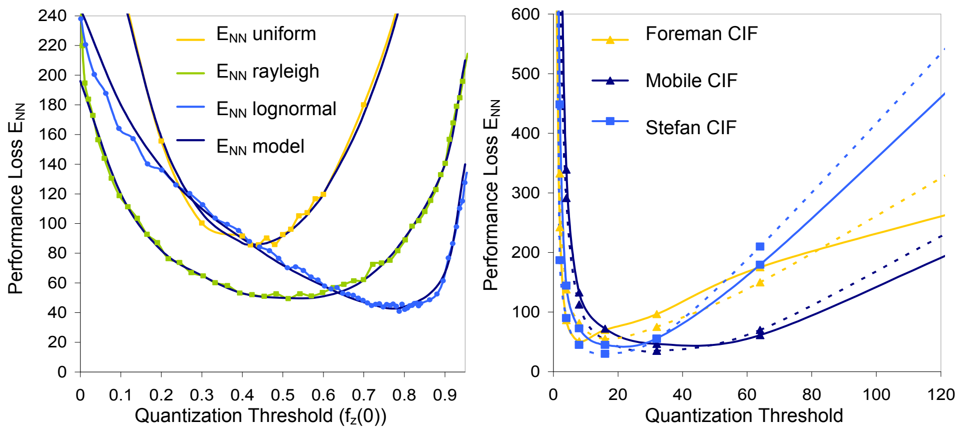


Figure 2.22: (Left) comparison of our objective function f_{obj1} simulated numerically with different input distribution settings. (Right) compares the objective function f_{obj1} based on the collected ME data (dashed lines) with simulated experiments excluding intra and skip modes (solid lines). Both used 1-bit quantization thus a single threshold (x-axis).

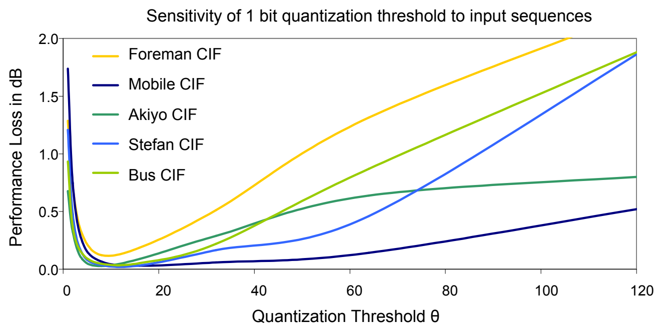


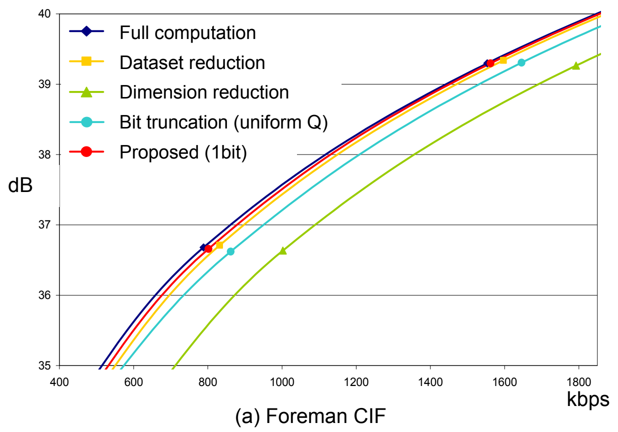
Figure 2.23: Illustration of performance degradation in coding efficiency with relation to the 1-bit quantizer threshold value for various input sequences. Representatively different test sequences (low/high motion, texture) were selected to cover a wide range of variations in input sequences. Result shows very low sensitivity to input sequence variation in terms of the optimal threshold value. This confirms high *homogeneity of viewpoints towards nearest-neighbors* assumption which our proposed algorithm is based on. The range of threshold is from 0 to 255 while only 0 to 120 range is shown.

compares our cost function based on the collected ME data with simulated experiments.

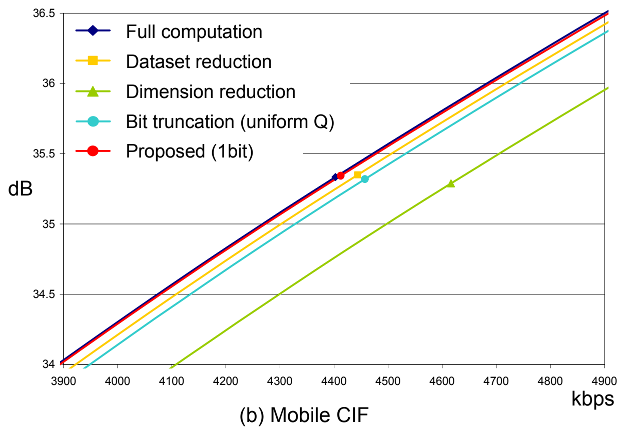
Fig. 2.23 provides some insight about the sensitivity of optimal threshold to input variation. Despite large variation of the input source characteristics, dimension-distances where quantization is applied exhibit more consistent statistical behavior, leading to overall robustness in our quantization method.

2.7.2 Vector Quantization for Data Compression

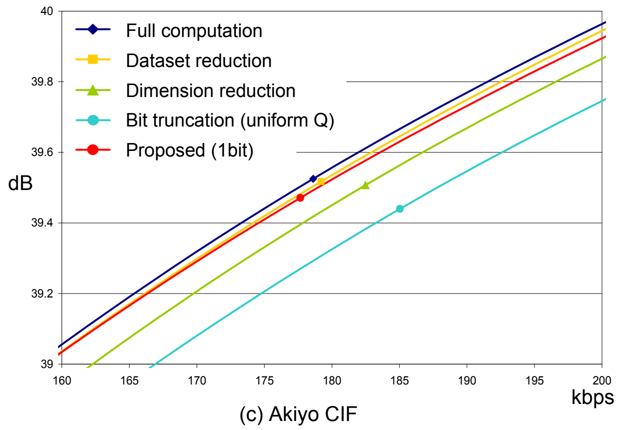
A vector quantizer encodes a multidimensional vector space into a finite set of values from a discrete subspace of lower dimension. A lower-space vector requires less storage space, so vector quantization (VQ) is often used for lossy compression of data such as image, video, audio or for speech recognition (statistical pattern recognition).



(a) Foreman CIF

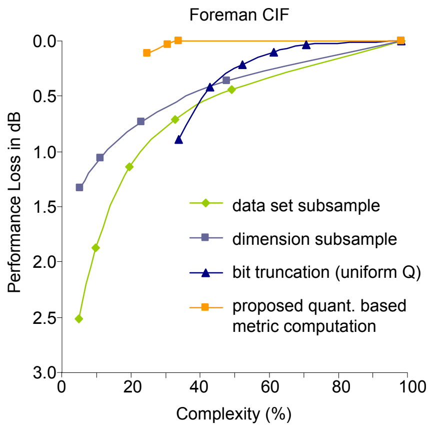


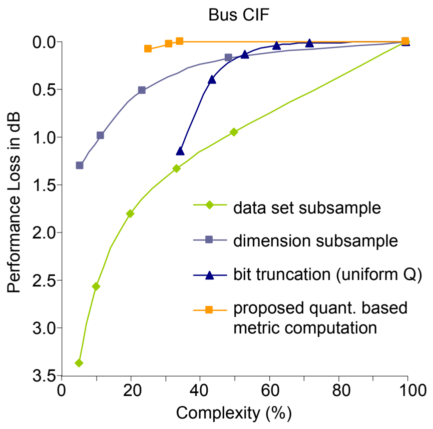
(b) Mobile CIF



(c) Akiyo CIF

Figure 2.24: Comparisons of video compression performance in rate distortion sense for five different motion estimation scenarios: i) conventional full search and full metric computation, ii) checking points are reduced, iii) dimensionality is reduced, iv) uniform quantization (bit truncation) is performed, and v) proposed 1 bit quantization based metric computation. scenarios ii), iii), and iv) have the same complexity reduction.





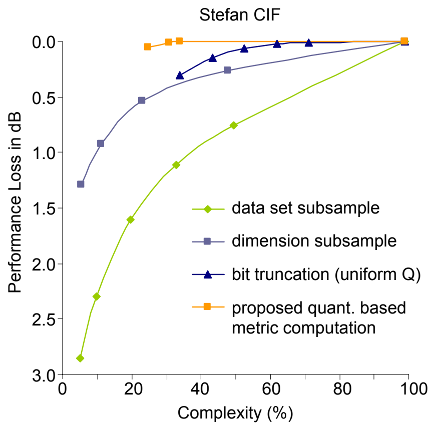


Figure 2.25: Comparisons of Complexity-Performance trade-offs of four different scenarios: each scenario reduces i) size of a data set R, ii) dimensionality of each data $\mathbf{r} \in R$, iii) bit depth of each data dimension by truncating least significant bits (equally seen as uniform quantization on each data dimension), and iv) resolution of each dimension-distance (proposed distance quantization based metric computation. X axis represents complexity percentage to that of original full computation. Y axis represents the RD performance loss measured in dB, thus zero represents no performance degradation.

VQ maps D-dimensional vectors in the vector space $\mathcal{R}^D$ into a finite set of vectors $Y = \{\mathbf{y}_i : i = 1, 2, , N\}$. Each vector $\mathbf{y}_i$ is called a code vector and the set of all the code vectors, Y, is called a codebook. A vector space $\mathcal{R}^D$ is partitioned into N code cells (clusters) $C = \{C_i : i = 1, 2, , N\}$ of non-overlapping regions where each region C_i called Voronoi region is associated with each codeword, $\mathbf{y}_i$. Each Voronoi region C_i is defined by:

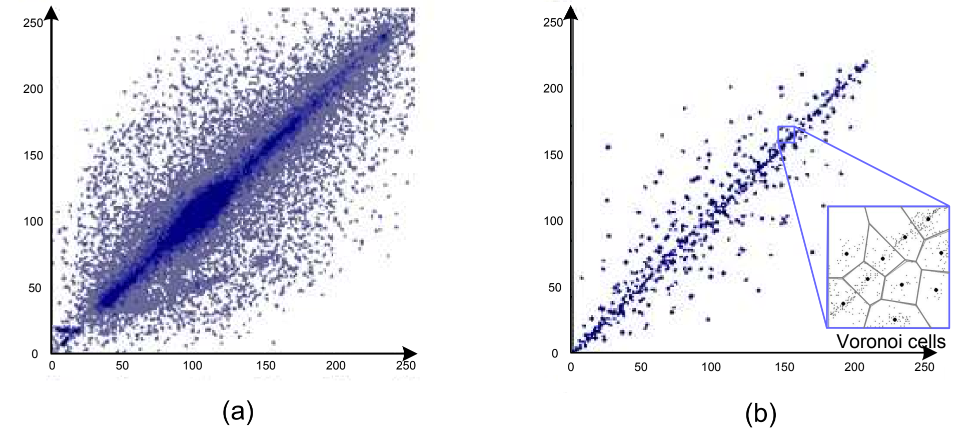
$$C_i = \{\mathbf{x} \in \mathcal{R}^D : \|\mathbf{x} - \mathbf{y}_i\| \leq \left\|\mathbf{x} - \mathbf{y}_j\right\|, \forall j \neq i\}$$

where VQ maps each input vector $\mathbf{x} \in C_i$ in code cell C_i to the code vector $\mathbf{y}_i$. Fig. 2.26(a) and (b) illustrate this in the case of a simple representation of a gray-scale image in a 2D vector space by taking in pairs the values of adjacent pixels (a) and their corresponding set of 512 code cells/Voronoi regions (b).

VQ consists of two process: designing a codebook and performing nearest code vector search. There are several VQ algorithms which have different codebook design method but all perform exact nearest neighbor search. In this section, we use well-known generalized Lloyd algorithm (GLA) to design a codebook and use QNNM metric to search that codebook.

VQ performs NNS for each input vector (a query) to all code vectors and encodes it with the nearest code vector. A set of all code vectors can be seen as data set of NNS problem (Fig. 2.26(b)). Fig. 2.26(c) and (d) each illustrate 2D example of distance distribution to all code vectors F_V and distance distribution to the nearest code vector F_{NN}, respectively.

Fig. 2.27 and Fig. 2.28 show 1-bit and 2-bit QNNM performance when it is applied to codebook search process of VQ based image coding. Fig. 2.27 compares four different VQ settings: (i) using VQ codebook obtained by generalized Lloyd



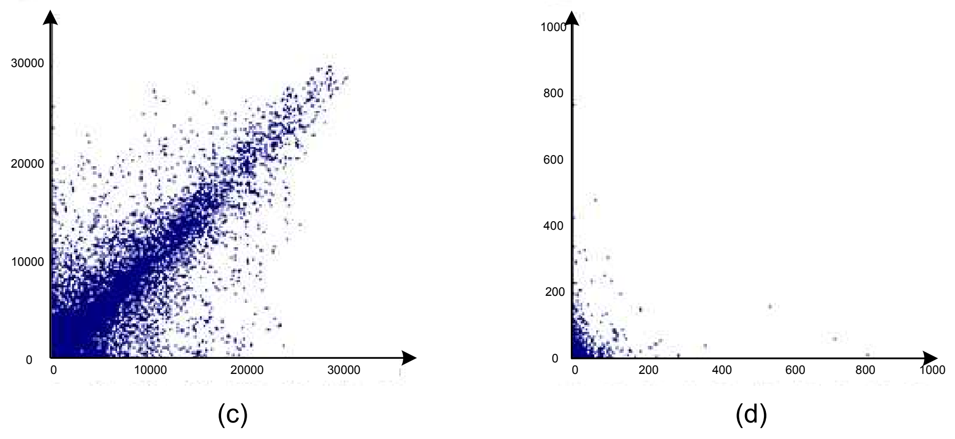


Figure 2.26: 2D statistical distribution of vector quantization data. (a) shows distribution of the input training image data to generate a codebook. (b) shows distribution of a set of code vectors (codebook with a size of 512) where NNS is performed with every query vector to find its closest code vector (F in metric space). (c) represents distribution of difference/distance from a query vector to every code vector (F_V in viewpoint space). (d) shows distribution of distance from a query vector to its NN/best matching codeword (F_{NN} in viewpoint space).

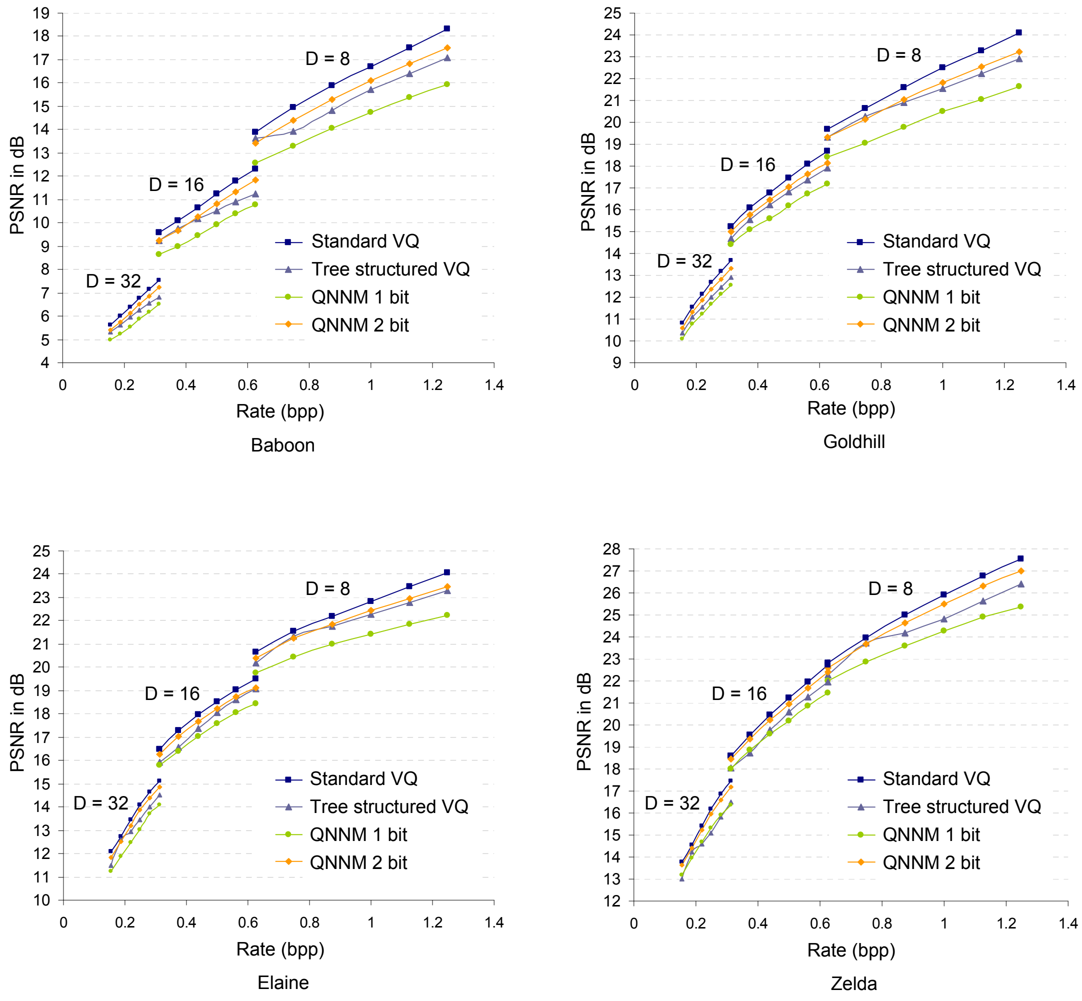


Figure 2.27: Rate-distortion curves of VQ based image coding performance for 4 gray-scale 512x512 test images. Four different VQ scenarios are compared: (i) generalized Lloyd algorithm (GLA) based VQ codebook with ℓ_2 distance (standard VQ), (ii) tree-structured vector quantization (TSVQ), (iii)(iv) GLA based VQ codebook with 1-bit QNNM and 2-bit QNNM. Performance was compared for dimensionality 8, 16, and 32 and for different codebook sizes, 32, 64, 128, 256, 512, 1024. 2-bit QNNM outperforms TSVQ method.

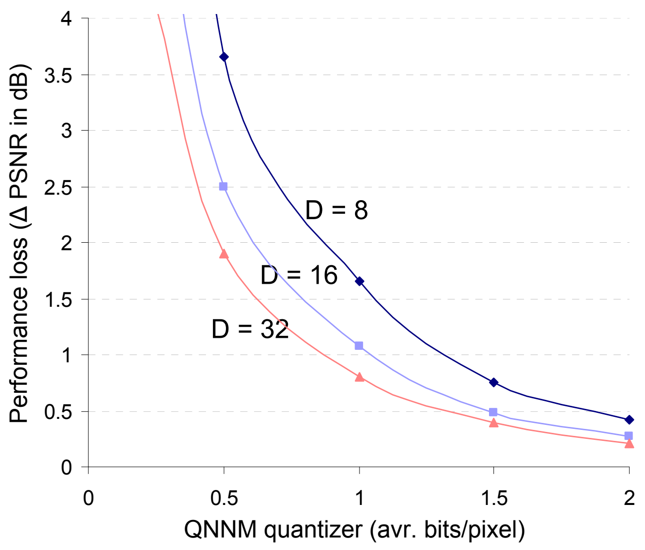
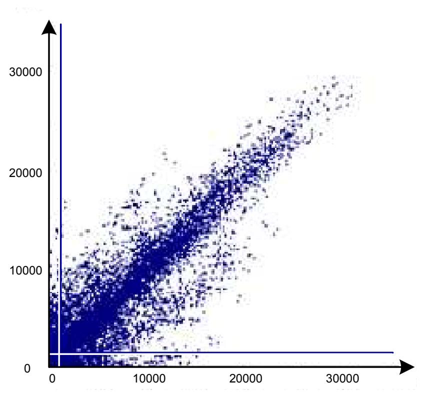


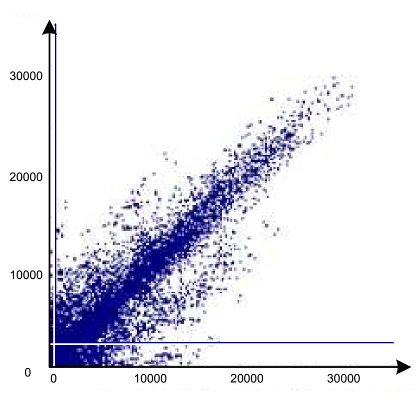
Figure 2.28: Average performance loss (average over different codebook sizes and test images in PSNR) by QNNM decreases as the number of quantization level increases. Right axis represents the average number of bits per dimension/pixel allowed for QNNM. 3 different vector sizes/dimensionality, 8, 16, and 32, were shown. This result as well as the one of ME application both support our previously drawn performance analysis that QNNM performance improves with dimensionality.

algorithm (GLA) and ℓ_2 distance search metric (standard VQ) (ii) tree-structured vector quantization (TSVQ), (iii) using GLA based VQ codebook with 1-bit QNNM, and (iv) using GLA based VQ codebook with 2-bit QNNM. Performance was compared for dimensionality 8, 16, and 32 and for different codebook sizes, 32, 64, 128, 256, 512, 1024. Fig. 2.28 shows how average performance of QNNM used for codebook search changes with dimensionality and average number of quantization levels per pixel. Both Fig. 2.27 and Fig. 2.28 numerically supports our analysis in Section 2.6.2 showing QNNM performance improves with dimensionality.

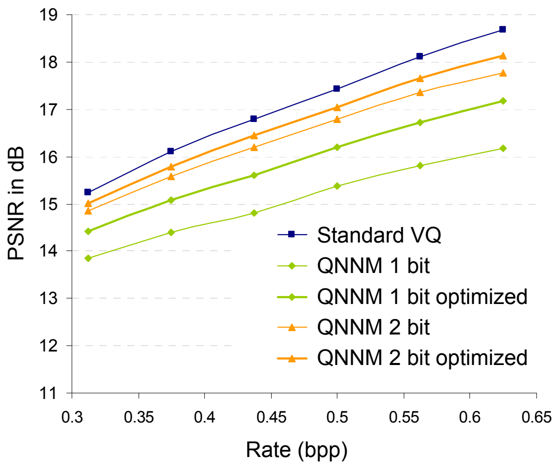
Similarly to ME application, input distribution to VQ and distribution of codebook F as well as distance distribution to code vectors F_V, all of these share similar statistical characteristics of high correlation and identical marginal distributions as shown in Fig. 2.26. This allows QNNM to exploit their correlation information to



(a)



(b)



(c)

Figure 2.29: While VQ input data is identically distributed on each dimension or pixel, their optimal **Q** thresholds are not identical in all dimensions. (a) and (b) show 2D examples of having (a) identical and (b) different **q** threshold for each dimension. Difference of optimal **Q** thresholds of different dimensions increases and performance improves as correlation between dimensions becomes stronger. (c) shows how much performance of VQ based image coding (goldhill with D=16) improves as **q** threshold is optimized in terms of exploiting correlation information between dimensions. It is also the case for ME application and others which involve data set with high correlation.

improve QNNM performance further. Fig. 2.29(a) and (b) illustrate **q** quantizer example using identical quantizer for all pixels (a) and different quantizer for different pixels (b) and show why (b) partitions viewpoint space better to approximate NN. [10] Fig. 2.29(c) show how much performance can be improved by exploiting correlation information. Thin curve line of QNNM represents QNNM with identical quantizer for all pixels while thick curve line represents optimized quantizers which take statistical correlation information into account.

In general, QNNM performance on VQ image coding results in more significant performance degradation than that of ME application. Therefore QNNM is not highly recommended for VQ application. This is not because of statistical characteristics of input data to VQ system but because NN search error in VQ introduce direct impact on compression efficiency. Also VQ for image coding in general use smaller blocks to code (e.g., 2×2 to 4×4). Unlike ME, ℓ_2 metric is also not an approximated metric of optimal complex metric (e.g., transform, quantization, entropy coding combined) but it is optimal itself in generating codebook and achieving the best compression given codebook. In other words, even if NN approximation error $\bar{\epsilon}$ is similar for both ME and VQ application, its impact on system performance is more tolerant with ME and more severe with VQ application. However, if QNNM is used in VQ for audio/voice data compression which often requires high dimensionality (e.g., $D > 500$), NN approximation error $\bar{\epsilon}$ could result in much more tolerant result.

[10] threshold values shown in Fig. 2.29(a)(b) do not represent the actual optimal threshold values which is much smaller than those of illustrated. Actual optimal threshold values are too small to be visible on graphs.

2.8 Conclusions and Future Work

This chapter introduced a novel methodology, called *quantization based nearest-neighbor-preserving metric approximation algorithm*(*QNNM*), which leads to significant complexity reduction in search metric computation. Proposed algorithm exploits four observations: (i) homogeneity of viewpoint property, (ii) concentration of extreme value distribution, (iii) NN-preserving not distance-preserving criterion, and (iv) fixed query information during search process. Based on these, *QNNM* approximates original/benchmark metric by applying query-dependent non-uniform quantization directly on the dataset, which is designed to minimize the average NNS error, while achieving significantly lower complexity, e.g., typically 1-bit quantizer. It entails nonlinear sensitivity to distance such that finer precision is maintained only where it is needed/important while unlikely regions to be nearest neighbors are very coarsely represented. We show how the optimal query adaptive quantizers minimizing NNS error can be designed off-line without prior knowledge of the query information to avoid the on-line overhead complexity and present an efficient and specifically tailored off-line optimization algorithm to find such optimal quantizer. Three distinguishing characteristics of *QNNM* are statistical modeling of dataset, employing quantization within a metric, and query-adaptive metric, all of which allow *QNNM* to improve performance complexity trade-off significantly and provide robust result even when the problem involves non-predefined or largely varying data set or metric function. With motion estimation application, QNNM with coarsest 1-bit quantizer per pixel (note that a quantizer for each pixel is different to exploit the correlation of input distribution) results in on average 0.01dB performance loss while reducing more than 70% to 98% metric computation cost.

Our proposed approach can be also used for k-nearest neighbor search or orthogonal range search. To find the exact nearest-neighbor, this approach can be also used as a preliminary filtering step to filter out unlikely candidates and then refinement process can be performed within the remaining set. This method can be performed independently but also in parallel with most of existing preprocessing based algorithms.

More work needs to be done in terms of specific logic/circuit level design for efficient hardware implementation of quantizer. Also it would be interesting to see how this method can be incorporated with parallel and distributed index structures. Furthermore, if possible, generalization of our proposed work can be developed to deal with more general metric function such as quadratic metric and relax the constraint we posed in this thesis on metric structure (no cross-interference among dimensions in metric function).

The concept of our proposed approach can be extended to numerous problems involving search process such as combinations of queries, batch queries, classification problems, other proximity problems etc. But not limited to similarity search problem, the potential benefit of error tolerance concept can be reaped from variety of application areas.

Chapter 3

Fault Effect Modeling for Nearest Neighbor Search Problem

In this chapter, we investigate the impact of hardware faults on NNS problem with a focus on NNS metric computation process. More specifically, we provide an analytical formulation of the impact of single and multiple stuck-at-faults within NNS metric computation. We further present a model for estimating the system-level performance degradation due to such faults, which can be used for the error tolerance based decision strategy of accepting a given faulty chip. We also show how different faults and NN search algorithms compare in terms of error tolerance and define the characteristics of search algorithm that lead to increased error tolerance. Finally, we show that different hardware architectures performing the same metric computation have different error tolerance characteristics and we present the optimal hardware architecture for NNS metric computation in terms of error tolerance. Our work could also applied to systems (e.g., classifiers, matching pursuits, vector quantization) where a selection is made among several alternatives (e.g., class label, basis function, quantization codeword) based on which choice minimizes an additive metric of interest.

3.1 Introduction

Error tolerance approach raises the threshold of conventional perfect/imperfect to acceptable/unacceptable by analyzing the system-level impacts of faults and accepting minor defects which result in slightly degraded performance within some application specific ranges of acceptability. However, a critical factor for this approach to be successful is to be able to cost-efficiently test and accurately predict if a defective chip will provide acceptable system-level performance without having to perform application level testing.

In previous work, the impact of hardware defects that lead to faults at circuit interconnects and soft errors produced by voltage scaling have been studied. Hardware faults such as those arising in a typical fabrication process can potentially lead to "hard" errors, since some of the functionality in the design is permanently impaired, whereas "soft" errors may arise when a circuit operates at a voltage lower than specified for the system. Previous work [17, 18] showed that certain range of hardware defects within the motion estimation (ME) and discrete cosine transform (DCT) with quantization subsystems lead to acceptable quality degradation. It also proposed a novel ET based testing strategy for such systems. Other recent work [9] showed that the ME process exhibits significant error tolerance in both hard and soft errors and further proposed simple error models to provide insights into what features in NNS algorithms lead to increased error tolerance.

In this chapter, based on the same ET concept, we provide an analytical formulation of the impact of multiple hardware faults on NNS metric computation process. This provides estimates of system-level performance degradation due to such faults, which can be used in deciding whether to accept a given faulty chip. Furthermore, based on this model, we investigate the error tolerance behavior of the

NNS process in the presence of multiple hardware faults from both an algorithmic and a hardware architecture point of view. In comparing different NNS algorithms we observe that high performance, low complexity search algorithms can in fact more error tolerant than other methods. As an example, in our experiments, enhanced predictive zonal search (EPZS) [14] algorithm for motion estimation process exhibits minimum degradation with respect to full search (FS) in the fault-free case (e.g., $0.01dB$ loss) but can perform significantly better in the presence of faults (e.g., up to $2.5dB$ gain compared to a faulty FS in some cases). When comparing different hardware architectures to perform the same metric computation, we also observe significant variations in error tolerance. We show that the optimal hardware architecture for NNS metric computation in terms of error tolerance is a perfectly balanced binary tree structure, which is also effective in terms of enabling parallel computation. Our simulations with motion estimation (ME) for video coding application show that, if the optimal structure is used, the expected error due to a fault can be reduced by up to 95%, as compared to other architectures, and that more than 99.2% of fault locations within metric computation circuits result in less than $0.01dB$ performance degradation.

Most previous research has relied on the single stuck-at (SA) fault assumption, which has been well-studied due to its simplicity and high fault coverage and has worked fairly well in practice. However, with decreasing feature sizes and increasingly aggressive design styles, single SA fault has become a rather restrictive model since it allows the defect to influence only one net, while defects in modern devices tend to cluster and affect multiple lines in the failing chip. Moreover, recent experiments [27] confirm that more than 40% of defects found in failing chips cannot be diagnosed using the single SA fault model. Multiple SA fault model on the other hand covers a greater percentage of physical defects, and can also be used

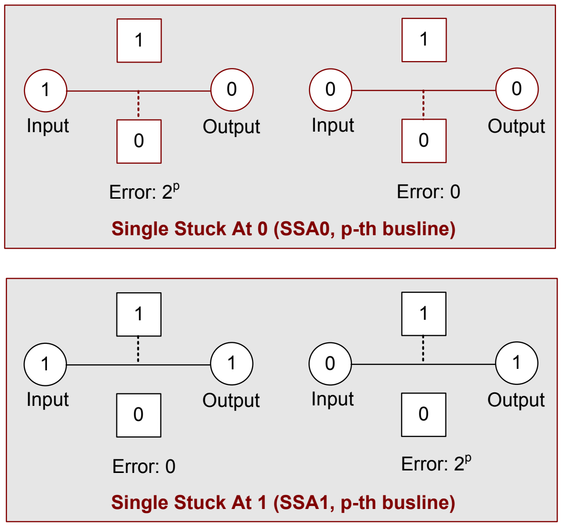


Figure 3.1: Simple illustration of single stuck at fault model.

to model defects of certain different fault types such as multiple bridging faults or multiple transition faults. However, the multiple SA fault case has not been studied extensively due to the issues of complexity and functional error inter-dependency characteristics. In this chapter, we present an analytical model of the system-level fault effect for any arbitrary multiple SA faults in a metric computation circuit and study error tolerance properties of the NNS process at both algorithmic and hardware architecture level based on multiple SA fault assumption.

The rest of this chapter is organized as follows. In Section 3.2, we begin with a brief description of several hardware architecture for metric computation process and SA fault model. In Section 3.3, we discuss appropriate measures for the impact of faulty hardware and provide simulation results using ME application. In Section 3.4, we formulate multiple SA fault effect on both metric computation level and matching process or search level. In Section 3.5, we analyze and define the characteristics of NNS algorithms that lead to increased error tolerance. In Section 3.6, we provide analysis on NNS metric hardware architecture pertaining to the error tolerance and show the optimal structure is perfectly balanced tree structure. Furthermore, we present experimental results and discussion on actual ET based decision example for motion estimation process application in the context of H.264/AVC, comparing different hardware architectures and search algorithms. Section 3.7 summarizes our conclusions and main results.

3.2 NNS Metric Computation Architecture & SA Fault Model

There are several types of hardware implementation architectures [42] for computing a NNS metric, with different levels of parallelism. We will refer to them as matching metric computation (MMC) architectures. Figure 3.2 illustrates a few examples of MMC architectures [42] which can be viewed as arrays of cascaded adders and represented as binary tree graphs, where each inner node represents an adder, an edge connecting two inner nodes represents a data bus, and a leaf node corresponds to the processing element (PE) that computes distance for each dimension (e.g., the pixel-level prediction error for ME application). Figure 3.2 and tree structured model of MMC architectures will be revisited in more detail when we discuss the optimal structure of MMC architecture in Section 3.6. Note that our fault effect model in Section 3.4 and analysis on the search algorithms in Section 3.5 are independent of the MMC architecture and valid for any hardware implementation structure used.

Throughout this chapter we consider only faults in the interconnect data bus that affect the data transfer between PEs within a metric computation hardware architecture. These interconnect faults are modeled with the stuck-at (SA) fault model, a well-known structural fault model that assumes that the design contains

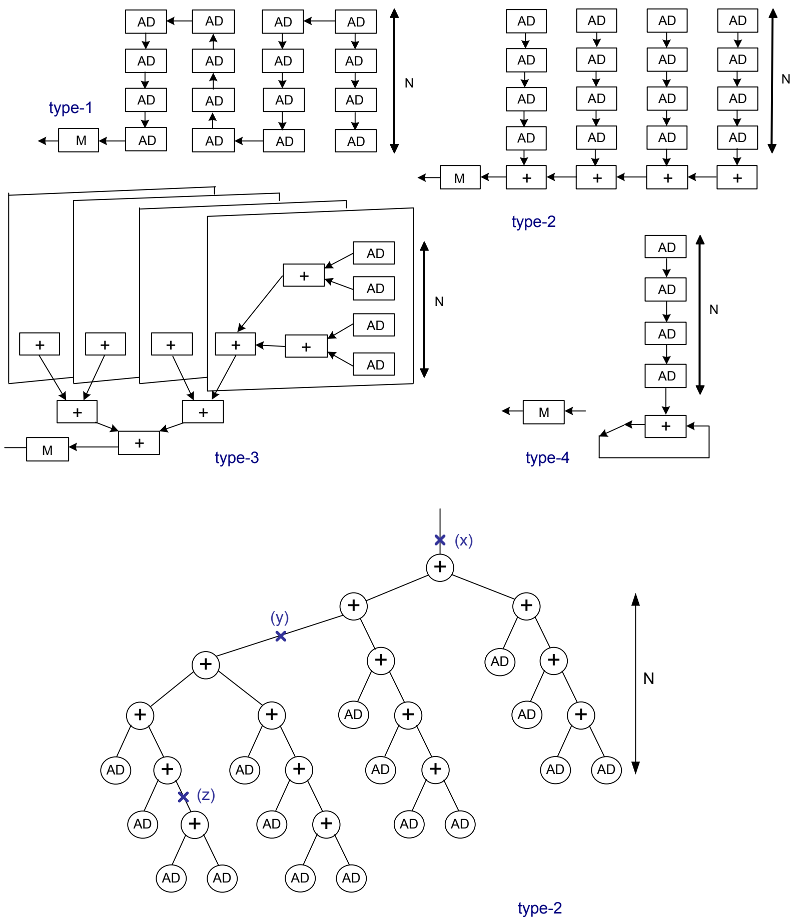


Figure 3.2: (Upper) Four examples of MMC architectures for NNS metric computation, represented as a dependence graph, where only processing elements (PEs) are shown for simplicity. AD denotes a processing element which is, for ℓ_1 metric for instance, an absolute difference and addition. M denotes a minimum value computation. (Lower) Tree structured flow graph corresponding to the type-2 MMC architecture on the left. In this graph, the processing element (leaf nodes, shown here as *AD*) and addition (inner nodes) computations are separated, thus AD denotes only the distance computation for each dimension (e.g., absolute difference for ℓ_1 metric). If ℓ_2 metric is used as a cost metric, leaf nodes become PEs computing a square difference.

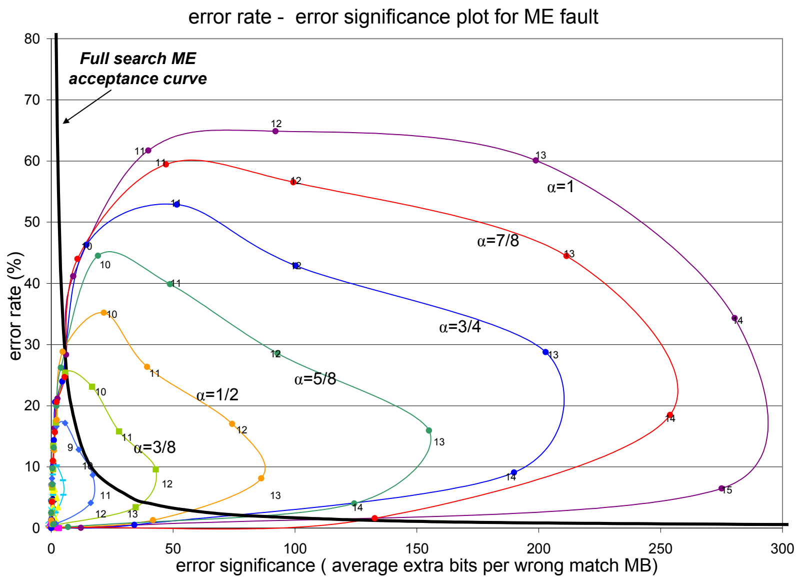


Figure 3.3: Example Error Rate and Significance due to SSA fault within MMCA for motion estimation application.

a fault that will cause a line in the circuit to behave as if it is permanently stuck at a logic value 0 (stuck-at-0 fault, abbreviated as SSA0) or 1 (stuck-at-1 fault, abbreviated as SSA1).

The SA fault model covers 80-90% of the possible manufacturing defects in CMOS circuits [50], such as missing features, source-drain shorts, diffusion contaminants, and metallization shorts, oxide pinholes, etc.

3.3 Quality Measure for Nearest Neighbor Search with Faulty Hardware

Faults in a MMC architecture would imply that it is likely that the selected point $\mathbf{r}_f$ may not be equal to the nearest neighbor point $\mathbf{r}^*$ which can be obtained with fault-free MMCA. We define NNS *Error* if $\mathbf{r}_f \neq \mathbf{r}^*$ due to a fault. An error does not occur for all queries but only occurs if certain conditions are met. An error occurs iff, a) if a fault is in the p-th data line, then the input to p must be 0 for $\mathbf{r}^*$ and 1 for $\mathbf{r}_f$, and b) $0 < d(\mathbf{r}_f, \mathbf{q}) - d(\mathbf{r}^*, \mathbf{q}) \leq 2^p$. Therefore, our focus is on how often these errors occur (error rate $P_e = prob(\mathbf{r}_f \neq \mathbf{r}^*)$) and how much additional quality degradation is introduced (error significance $S_e = d(\mathbf{r}_f, \mathbf{q}) - d(\mathbf{r}^*, \mathbf{q})$, representing the level of inaccuracy of MCP). Error rate and significance depend highly on the fault position with a certain variation due to the input sequence characteristics. Figure 3.3 demonstrates clearly how error rate and significance values change with faults at different positions. Points connected with the same line are faults in the same interconnect data bus with a different data bit line. Points shown in outer lines indicate faults in the data bus which has 32 more leaf nodes connecting towards that data bus than the adjacent inner line. Also note that SSA0 and SSA1 faults at the same positions produce identical results in both error rate and significance. Proof of this and further analysis on this concept of error rate and significance of hardware fault are provided in [10].

We further discuss fault error measures using more specific example application, comparing DCT process and ME process for video coding. While faults introduced within the DCT block within video coding tend to have a direct impact on visual quality degradation and the type of artifacts introduced, faults on ME metric have a more indirect impact on overall quality by impairing the accuracy of the ME engine. Thus, degradation of motion compensated (MC) prediction signal and its increased residual energy will be compensated with the rate penalty with no significant visual quality loss if Rate Control (RC) is not employed. However, for most cases, there are constraints imposed on the bit rate of the system where a certain local average bit rate needs to be maintained over time. Considering that most modern video encoders use RC schemes, additional bits required by increased residual of a block/frame, given certain bit-rate constraint, will be compensated by

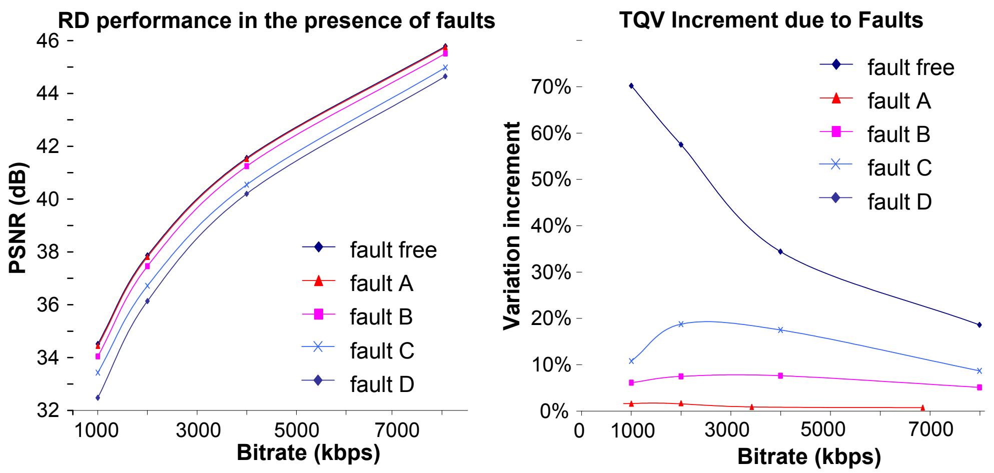


Figure 3.4: (Left) Rate-Distortion impact of faults. (Right) Temporal Quality Variations due to Faults.

using fewer bits for other blocks/frames. For example, assuming that a block level RC model is used, the increase in bit-rate due to faulty ME could imply increase in the quantization scale chosen to encode the subsequent blocks. In this case, artifacts would appear in the form of standard quantization artifacts, rather than, as was the case for DCT, error-specific artifacts. Therefore, the problem of measuring the visual quality degradation due to fault introduction within the ME can be also seen as the problem of measuring the impact on picture quality of a compression process. Thus, any visual quality metric which captures the video compression impairment reliably can be also applied to this case.

For example, most commonly accepted and widely used visual quality metric is MSE, or equivalently PSNR which simply computes distortion energy thus it lacks of correlating well with other important characteristics of human visual system. However, several experiments with human interference evaluated PSNR as a competitive metric for compression distortion assessment [2].

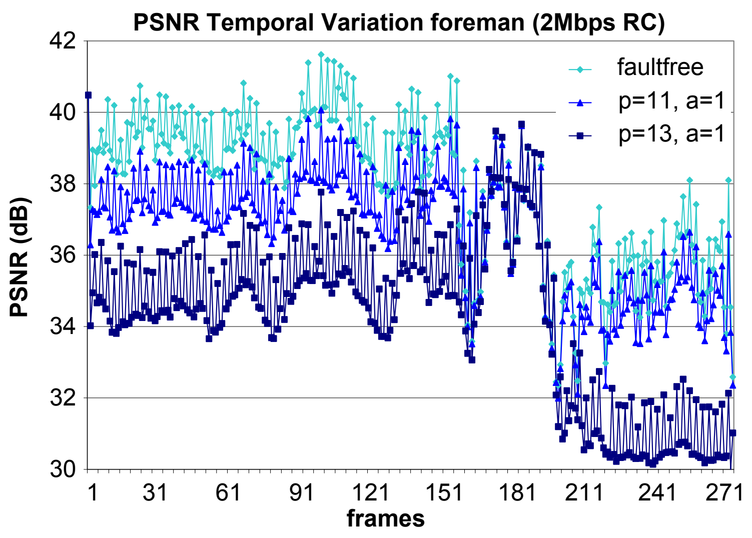


Figure 3.5: Rate control scheme distributes the fault impact over time. a SSA fault within the ME metric computation circuit is simulated.

Another problem of the average PSNR or MSE of all frames in a video sequence to represent quality degradation is that it can easily underestimate distortion variation throughout the sequence. In reality [2], an end-user will evaluate an entire video sequence based upon the quality variations and minimum quality.

Therefore, distortion variation can be separately considered into the spatial variation, a block level quality variation within a frame, and temporal variation, a frame level one throughout the sequence. This distortion variation measure can be particularly meaningful for the study of ME faults since we observed that introducing a SSA fault in ME always increases both the temporal and spatial quality variation on the video output. Note that the level of variation increase depends on the RC scheme. Typically RC performs bit allocation by selecting the encoder's quantization step size (QP) for the residual block/image. Note that most modern implementations of RC, a constraint is employed on the increment/decrement of the quantizer, which results in distributing distortion throughout the picture. Therefore errors occurring with certain rate are subdued and smoothly spread out over the picture after the rate controlled quantization process. Consequently, distinguishing errors into two measures of error rate and significance becomes no longer necessary. We have observed that spatial distortion variations tend to be imperceptible, on the other hand temporal variations can be significant. Spatial variation was measured by computing the variance of Qp values for each frame and by averaging them. To evaluate temporal variations within a video sequence we defined the measure TQV as:

$$TQV = \frac{\sum_{i=0}^{N-2} \left| MSE_i^{fault} - MSE_{i+1}^{fault} \right|}{\sum_{i=0}^{N-2} \left| MSE_i^{no\text{-}fault} - MSE_{i+1}^{no\text{-}fault} \right|} \tag{3.1}$$

where MSE_i^{fault} and $MSE_i^{no\text{-}fault}$ are the frame MSE values of the decoded images with and without ME faults respectively, and N is the total number of frames considered.

In Figure 3.4 (left) we present the Rate Distortion performance for the Foreman sequence, at CIF resolution, in the presence of ME faults within an MPEG-2 encoder. Similarly,the Temporal Quality Variation for the same faults are presented in Figure 3.4 (Right). Since error rate and significance measurements are no longer useful after RC Quantization for ME fault case, PSNR with additional measure of temporal quality variation would be able to represent well the quality degradation introduced by ME fault. However, we observed that temporal variation increase is relatively small compared to PSNR change and roughly proportional to the PSNR degradation. Thus it is well captured by PSNR measure. Figures 3.4 (Left) and 3.4 (Right) illustrate well this point. Large quality variations in Figure 3.4 (Right) are also related to a rather significant drop in PSNR in Figure 3.4 (Left). Therefore, for the evaluation of ME faults, PSNR can still capture most of quality impairment sufficiently although temporal variation metric could potentially improve the accuracy of visual quality assessment. In Figure 3.3, a PSNR based quality threshold

T (e.g. $T = 0.1dB$) was considered to classify faults. This threshold essentially defines an *acceptance curve*, according to which faults below this curve are considered as acceptable, while faults above are considered as unacceptable.

3.4 Multiple Fault Effect Modeling

In this section we present a complete analysis of the effect of interconnect faults in NNS process. We start by providing a model to capture the effect of any fault (or combination of multiple faults) on NNS metric. We then describe how these errors in metric computation lead to errors in the matching/searching process. Our motivation is twofold. First, we will use this analysis to predict the behavior of NNS algorithms and MMC architectures in the presence of faults (see Sections 3.5 and 3.6, respectively). Second, this analysis is a required step towards developing comprehensive testing techniques to identify acceptable error behavior. In particular, it will provide tools to tackle multiple fault scenarios, by discarding specific sets of faults (e.g., when one of the multiple faults is by itself unacceptable), establishing equivalence classes across faults (e.g., cases when dependent and independent faults may have the same impact), or determining how the parameters of the faults involved affect overall behavior (e.g., multiple faults with similar parameters may lead to relatively worse errors than sets of faults with differing characteristics).

Faults in a MMC architecture can lead to errors in the cost metric computation. We will refer to this as a metric computation (MC) error and denote its magnitude as $\Delta(D, \{f_i\}) = \hat{D} - D$ where $\hat{D}$ and D denote the computed costs with and without MC error and f_i denotes a fault. In general $\Delta(D, \{f_i\})$ is a function of the input sequence and the characteristics of the faults (e.g., their types, locations, and their dependencies). However, to simplify the notation, from here on we omit

$\{f_i\}$, a set of parameter vectors representing each fault's characteristics. We will refer to this function $\Delta(D)$ as the *MC error function*. Note that MC errors do not necessarily lead to matching process (MP) errors. We say a block suffers a MP error if $\mathbf{r}_f \neq \mathbf{r}^*$ where $\mathbf{r}_f$ and $\mathbf{r}^*$ represent the selected object/point and minimum object/point from given data set R, respectively . Since MP errors do not occur for all queries, we also define the MP error rate ($P_E = prob(\mathbf{r}_f \neq \mathbf{r}^*)$) which represents how often these MP errors occur, and the MP error significance ($S_E = d(\mathbf{r}_f, \mathbf{q}) - d(\mathbf{r}^*, \mathbf{q})$, where $d(\mathbf{r}_f, \mathbf{q})$ and $d(\mathbf{r}^*, \mathbf{q})$ are the computed distance from a query point to $\mathbf{r}_f$ and $\mathbf{r}^*$, respectively).

3.4.1 Binary Adder Tree and Fault Characterization

Each SA fault in any MMC architecture can be fully characterized by three attributes, a) the fault type $t \in \{0, 1\}$ (Stuck-At-0/SA0 or Stuck-At-1/SA1), b) the bit position of the faulty data line $p \in \{0, \cdots, P\}$ where 0 and P correspond to the LSB and MSB of faulty data bus respectively, depending on the MMC architecture, and c) the position of that data bus. The last attribute can be parameterized as a ratio $\alpha \in (0, 1\,]$ of the number of leaves in the subtree rooted at a faulty edge (data bus) to that of the entire tree. For example in Figure 3.2 (Lower), if a fault is positioned at (x), its corresponding α, say $\alpha_{(x)}$ is 1. Similarly, $\alpha_{(y)} = 1/2$ and $\alpha_{(z)} = 1/8$ and so forth. Therefore each fault i can be represented as $f_i = (t_i, p_i, \alpha_i)$, where $t_i \in \{0, 1\}$, $p_i \in \{0, \cdots, P\}$ and $\alpha_i \in (0, 1\,]$. Note that these fault parameters completely capture the fault effect, so that two faults with the same parameters in two different architectures have, on average, the same effect.

3.4.2 Multiple Fault Effect Modeling for Nearest Neighbor Search Metric Computation

Consider an interconnect fault i affecting a data bus. If the input to the data bus is x then we can represent the effect of the fault as a function $h_i(x) \triangleq \hat{x} - x$, where $\hat{x}$ is the output of the faulty data bus. For fault parameters p_i and t_i the shift induced by SA fault i can then be described as

$$h_i(x) = \hat{x} - x = (-1)^{t_i+1} \cdot 2^{p_i} \cdot I_{w^i_{t_i}}(x), \quad \text{where}$$

$$w^i_0 = \bigcup_{k \in 2N_0+1} [\, k \cdot 2^{p_i}, (k+1) \cdot 2^{p_i} \,) \quad \text{and} \quad w^i_1 = \bigcup_{k \in 2N_0} [\, k \cdot 2^{p_i}, (k+1) \cdot 2^{p_i} \,),$$

where N_0 denotes the set of non-negative integers. Note that $h_i(x)$ is a function of x, i.e., an intermediate term in the computation of the final matching metric cost D. Thus, modeling the final observed metric computation error would in principle require modeling the distribution of intermediate values x for a given D. Instead, as a simplification, we assume that, on average and for a given α_i and D, we will tend to have $x \cong \alpha D$, i.e., the intermediate metric value x at the fault position is proportional to the number of pixel-errors accumulated up to that position[1]. With this approximation we define the error at the output corresponding to fault i as $H_i(D) \cong h_i(\alpha D)$, which can be written as:

$$H_i(D) \triangleq (-1)^{t_i+1} \cdot 2^{p_i} \cdot I_{W^i_{t_i}}(D), \tag{3.2}$$

[1]This would hold if pixel-errors were iid with mean dependent on D. However, based on the generalized central limit theorem even if there exist some weak dependencies between pixel-errors, this still holds true if none of pixel-error variables exert a much larger influence than the others.

where

$$W_0^i = \bigcup_{k \in 2N_0+1} \left[\frac{k \cdot 2^{p_i}}{\alpha_i}, \frac{(k+1) \cdot 2^{p_i}}{\alpha_i} \right) \quad \text{and} \quad W_1^i = \bigcup_{k \in 2N_0} \left[\frac{k \cdot 2^{p_i}}{\alpha_i}, \frac{(k+1) \cdot 2^{p_i}}{\alpha_i} \right)$$

$H_i(D)$ is a periodic piecewise constant square wave function of D taking values in the set $R_i = \{0, \quad 2^{p_i} \cdot (-1)^{t_i+1}\}$ with period $T_i = 2^{p_i+1}/\alpha_i$. If there is a single SA fault $f_i = (t_i, p_i, \alpha_i)$ in a MMC architecture, then with the above approximation the error at the output is $\Delta(D) \cong H_i(D)$, which leads to D being shifted by $\pm 2^p$ or remaining unchanged depending on D and f_i.

However, when we consider multiple SA faults, $\Delta(D)$ cannot be accurately modeled unless extra information regarding the dependency relation between faults is also provided. In other words, two sets of faults with identical fault configurations, $\{f_i\}_{i=1}^{M}$ with $f_i = (t_i, p_i, \alpha_i)$ could produce different MC error function $\Delta(D)$ depending on the dependency relation between faults. Figure 3.7 illustrates simple examples of multiple faults with different dependencies. If faults are placed serially in the same computation path, the output of previous faults along the path could affect the input to the following fault position. We refer to such faults as *dependent* faults (Figure 3.7 (a)). Note that the relative position of the faults in the architecture matters, i.e., the computations closer to the root of the tree are affected by the errors introduced closer to the leaves, but not the other way around. On the other hand, if faults are placed in separate paths such that the output of each fault position is independent from one another, we refer to them as *independent* faults (Figure 3.7 (b)). Figure 3.7 (c) shows the case of a simple combination of both independent and dependent faults. For each of these three scenarios, Figure 3.7 provides an example of multiple faults locations in the MMC hardware architecture represented as a binary tree and a fault dependency graph illustrating their

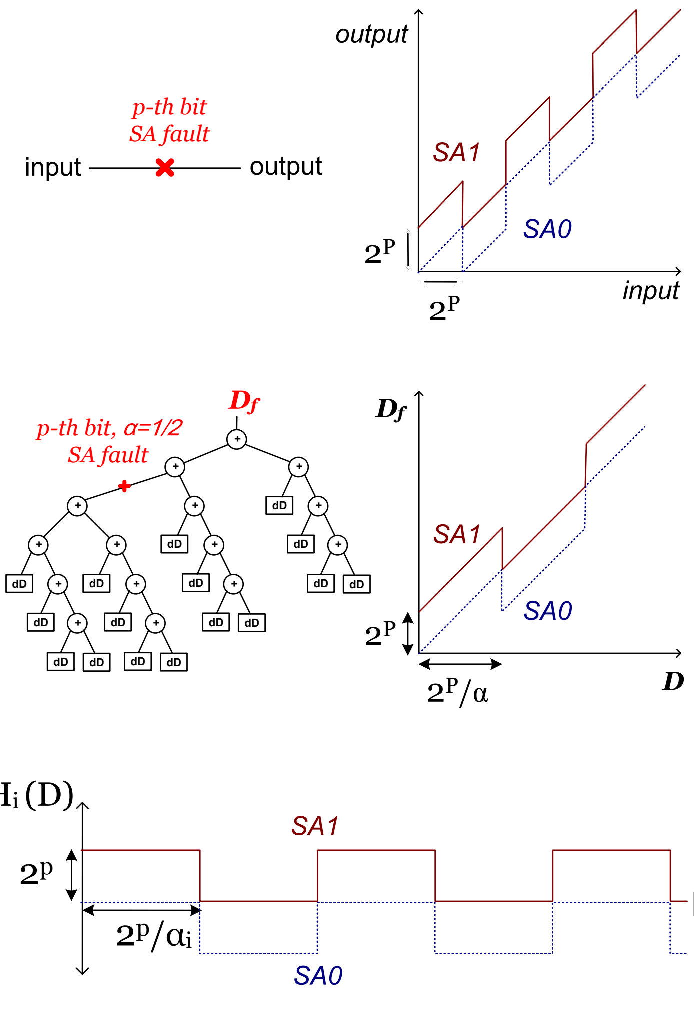
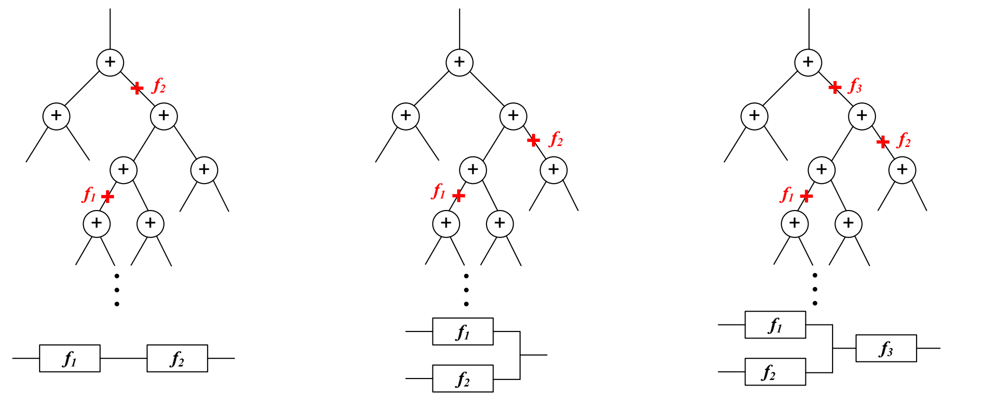


Figure 3.6: Illustration of single stuck at fault and its impact on output of that faulty interconnect (upper), and on output of final matching metric cost (lower).



$\Delta(D) = H_1(D) \odot H_2(D)$
(a) Dependent faults

$\Delta(D) = H_1(D) + H_2(D)$
(b) Independent faults

$\Delta(D) = (H_1(D) + H_2(D)) \odot H_3(D)$
(c) Combined faults

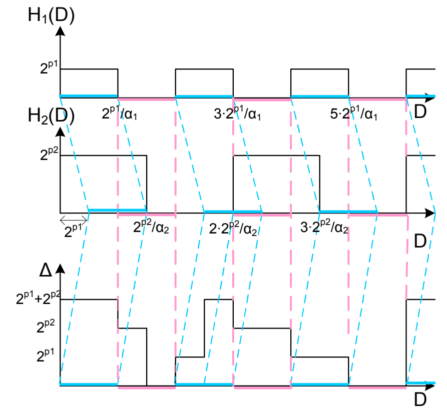
Figure 3.7: Examples of multiple SA fault cases with different dependency relations between faults. (a) two serially placed faults in the same path of circuit, (b) two parallel faults placed in separate paths, and (c) simple combination of (a) and (b) cases. These relations are illustrated more clearly in the fault dependency graphs shown below the binary trees. Metric computation (MC) error function $\Delta(D)$ can be formulated via functions of $\{H_i(D)\}_{i=1}^{M}$ defined in (3.2) using two basic operators $\{\odot, +\}$ (function addition and a variant of function composition operator defined in (3.3)), according to the dependency relation of a given set of faults as shown above, where D denotes the final cost value (e.g., SAD).

dependency relation. Similarly, any dependency relation of an arbitrary set of faults $S = \{f_i\}_{i=1}^{M}$ in a MMC architecture, where $f_i = (t_i, p_i, \alpha_i)$, can be represented as a combination of parallel and serial cascade of faults.

In order to formulate $\Delta(D)$ for any given set of faults with arbitrary fault dependency relations, we first introduce $(\mathcal{F}, +, \odot)$, an algebraic structure where the elements of $\mathcal{F}$ are periodic piecewise constant functions which are closed under two operations $\{+, \odot\}$. The operator $+$ denotes linear function addition and $\odot$ denotes a function composition operator defined as

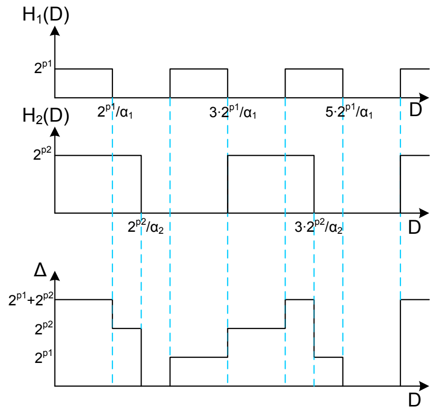
$$f(x) \odot g(x) \triangleq g(f(x) + x) + f(x) \tag{3.3}$$

This operator $\odot$ combines $f(x)$ and $g(x)$, where the input to $g(x)$ depends on the output of $f(x)$. These two operators $\{+, \odot\}$ essentially combine two MC error functions for two different sets of faults into one function providing the total cost shift incurred by both sets of faults. If two sets of faults are independent, the overall error function is the sum of the two error functions corresponding to each set of faults, thus the $+$ operator is used. Similarly, the $\odot$ operator is applied when two sets of faults are dependent. Figure 3.7 provides simple examples of multiple faults and their corresponding $\Delta(D)$ representation using two operators $\{+, \odot\}$. Figure 3.8 illustrates more graphically for each of the three cases discussed in Figure 3.7 how $\Delta(D)$ function is linked to the dependency relation of multiple faults. When two sets of faults are independent as in Figure 3.8 (b), $\Delta(D)$ function becomes linearly additive. On the other hand, when they are dependent, as in Figure 3.8 (a)(c), the error function will have to be computed by using the $\odot$ operator.



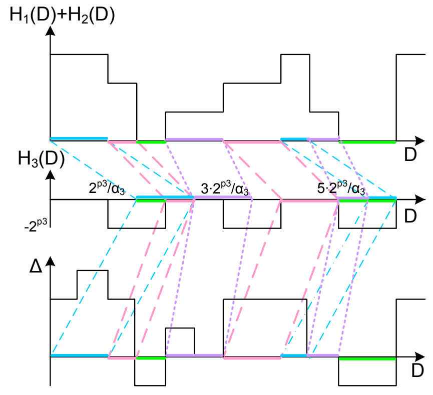
$\Delta(D) = H_1(D) \odot H_2(D) = H_2(H_1(D) + D) + H_1(D)$

(a) Dependent faults: f_1 and f_2



$\Delta(D) = H_1(D) + H_2(D)$

(b) Independent faults: f_1 and f_2



$\Delta(D) = (H_1(D) + H_2(D)) \odot H_3(D)$
$= H_3(H_1(D) + H_2(D) + D) + H_1(D) + H_2(D)$

(c) Combined faults: f_1, f_2 and f_3

Figure 3.8: $\Delta(D)$ computation for each of the three cases described in Figure 3.7. Faults f_1 and f_2 are SA1 faults with parameters $\{1, p_1, \alpha_1\}$, $\{1, p_2, \alpha_2\}$ respectively and f_3 is SA0 fault with $\{0, p_3, \alpha_3\}$. (a) a fault f_1 affects the input to the next fault position f_2. Thus, the total cost shift at D (MC error function $\Delta(D)$) is the summation of the cost shift due to f_1 at D, $(H_1(D))$ and that of f_2 at shifted D by $H_1(D)$, which is $(H_2(H_1(D) + D))$. (b)$\Delta(D)$ is the simple linear summation of $H_1(D)$ and $H_2(D)$. (c) only dependent relation between faults $\{f_1, f_2\}$ and f_3 is depicted which is essentially the same process as (a).

More formally, the MC error function $\Delta(D)$ of an arbitrary set of faults $S = \{f_i\}_{i=1}^{M}$ in any MMC architecture, where $f_i = (t_i, p_i, \alpha_i)$, with any dependency relation can be estimated using operators $\{+, \odot\}$ on the set of functions $\{H_i(D)\}_{i=1}^{M}$ resulting also in a periodic piecewise constant function of cost D, taking 2^M different possible values in a finite set R_S with period T_S, where[2]

$$R_S = \left\{ \sum_{i\in\mathcal{A}} 2^{p_i} \cdot (-1)^{t_i+1} \right\}_{\mathcal{A}\in P(\{1,\cdots,M\})} \qquad T_S = lcm\left\{ T_i = \frac{2^{p_i+1}}{\alpha_i} \right\}_{i=1}^{M} \tag{3.4}$$

Since the $\odot$ operator is non-commutative and non-distributive over addition, order is important in computing $\Delta(D)$. Based on its fault dependency graph (examples are shown in Figure 3.7 which can be also represented as a tree in which each node i correspond to a fault i), $\Delta(D)$ need to be updated iteratively at each node i of the tree from the leaves towards the root by combining each $H_i(D)$. The combination process at each node i is performed using $+$ and $\odot$ operators to combine siblings ($\{\Delta_j(D)\}_{j\in\mathcal{J}_i}$) and children ($\sum_{j\in\mathcal{J}_i} \Delta_j(D)$) with parent ($H_i(D)$), respectively, where $\Delta_i(D)$ represents the updated $\Delta(D)$ for a subtree rooted at node i and $\mathcal{J}_i$ is a set of all child nodes of node i.

$$\Delta_i(D) = (\sum_{j\in\mathcal{J}_i} \Delta_j(D)) \odot H_i(D)$$

Therefore for a given arbitrary set of multiple SA faults, we can obtain MC error function $\Delta(D)$ which indicates whether there is a MC error ($\Delta \neq 0$) and how much shift occurred due to faults ($\Delta = \hat{D} - D$) for any given final cost value D. Note that once we obtain MC error function ($\Delta(D)$), the dependency relation does not have to be reconsidered afterwards.

[2] $P(\cdot)$ and $lcm(\cdot)$ denote the power set and the least common multiple, respectively.

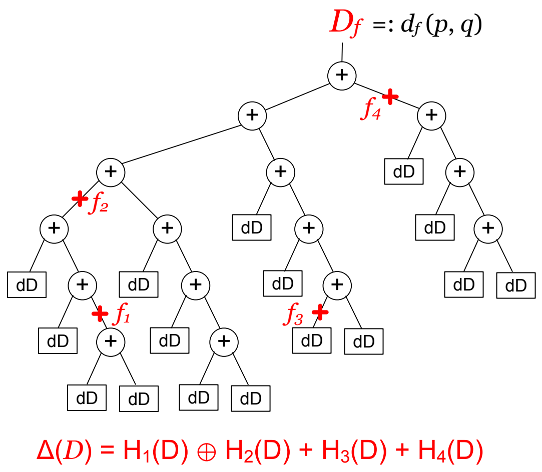


Figure 3.9: Simple example of MMC circuit with multiple faults and its corresponding MC error function.

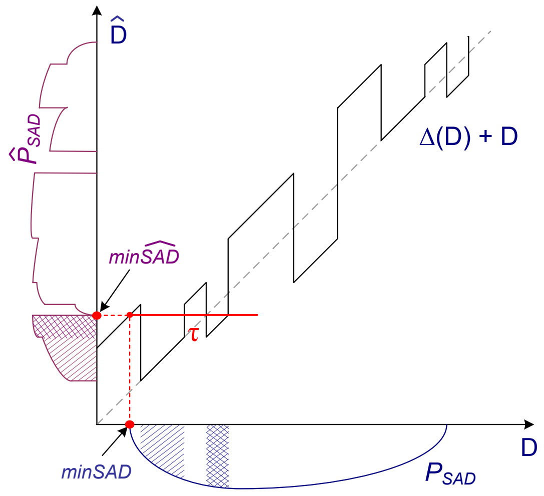


Figure 3.10: Analysis of matching process error due to multiple SA faults.

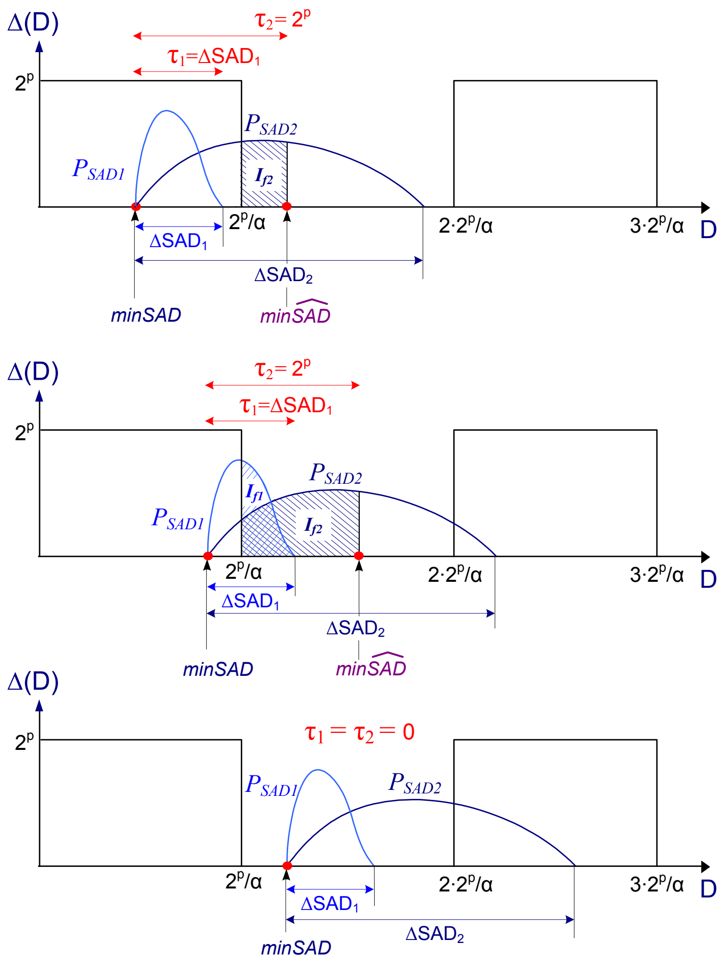


Figure 3.11: A comparison of two different sets of MV candidates with respect to the MP error due to a single SA fault.

3.4.3 Multiple Faults Effect Modeling for Matching Process

SA faults alter the computed cost values $\hat{D} = D + \Delta(D)$ according to the MC error function which we have analytically formulated in the previous section. However, this does not necessarily lead to matching process (MP) errors, which is the case where the selected candidate $\mathbf{r}_f$ based on the altered cost value $\hat{D}$ is not equal to the best $\mathbf{r}^*$. A MP error occurs if and only if, $D(\mathbf{r}_f) > D(\mathbf{r}^*)$ and $\hat{D}(\mathbf{r}_f) < \hat{D}(\mathbf{r}^*)$.

In this section, based on the MC error function $\Delta(D)$ we model the MP error with two assumptions: i) minimum distance follows a distribution P_{minSAD} which has different characteristics for different classes of video sequences (e.g., low motion vs. high motion video), and ii) the set of candidates N to be tested for the matching process can be modeled as N iid samples drawn from a distribution P_{SAD}. Based on these two assumptions, for a given $\Delta(D)$, MP error rate P_E and the expected MP error significance $\bar{E}$ are formulated as follows.

SA faults shift all computed costs by $\Delta(D)$ which in turn alter P_{SAD} into $\hat{P}_{SAD}$ where

$$\hat{P}_{SAD}(\hat{D}) = \sum_{\forall D: D+\Delta(D)=\hat{D}} P_{SAD}(D) \tag{3.5}$$

Figure 3.10 (left) illustrates an example of how a given P_{SAD} is mapped into $\hat{P}_{SAD}$ using $\Delta(D)$ function. MP error occurs if the shifted minimum SAD ($min\widehat{SAD}$) is not the minimum of $\hat{P}_{SAD}$ in $\hat{D}$ domain and if there exists at least one candidate that falls within the shaded region shown in Figure 3.10. We refer to this shaded area as *error region* which essentially indicates the range of SAD values satisfying $SAD > minSAD$ and $\widehat{SAD} < min\widehat{SAD}$ conditions and therefore determined by

$\Delta(D)$. This *error region* is bounded by *error region bounding interval* τ which is a function of $minSAD$ value and defined as

$$\tau(D) \triangleq \Delta(D) - minR_S \tag{3.6}$$

where $minR_S$ denotes the minimum of a set R_S defined in (3.4). We denote $\hat{F}_{SAD}$ as a cumulative distribution function of $\hat{P}_{SAD}$ and define a new function $\xi_N(D) \triangleq (1-\hat{F}_{SAD}(\hat{D}))^N$ that gives the probability that shifted cost values of all N candidates fall higher than $\hat{D}$. Then we can represent MP error rate for a given minimum distance value (denoted as D_{min}) as $P_{E|D_{min}} = 1 - \xi_N(D_{min})$. Therefore,

$$P_E = \sum_D P_{minSAD}(D) \cdot P_{E|D} = \sum_D P_{minSAD}(D) \cdot (1 - \xi_N(D)) \tag{3.7}$$

Similarly, expected value of MP error significance for a given D_{min} can be represented as

$$\mathbb{E}_{D_{min}}(S_E) = \sum_{d \in \mathcal{X}_{D_{min}}} (d - D_{min}) \cdot \xi_{N-1}(d) \cdot P_{SAD}(d)$$

where $\mathcal{X}_{D_{min}} = \left\{ d \mid D_{min} < d < D_{min} + \tau(d),\ \hat{d} < \hat{D}_{min} \right\}$ is a set of d corresponding to the blue shaded area of $P_{SAD}(d)$ in Figure 3.10 (Left).

Similarly, the expected MP error can be expressed as

$$\bar{E} = \mathbb{E}(S_E) = \sum_D P_{minSAD}(D) \cdot \mathbb{E}_D(S_E)$$

$$\bar{E} = \mathbb{E}(S_E) = \sum_D P_{minSAD}(D) \sum_{d \in \mathcal{X}_D} (d - D) \cdot \xi_{N-1}(d) \cdot P_{SAD}(d) \tag{3.8}$$

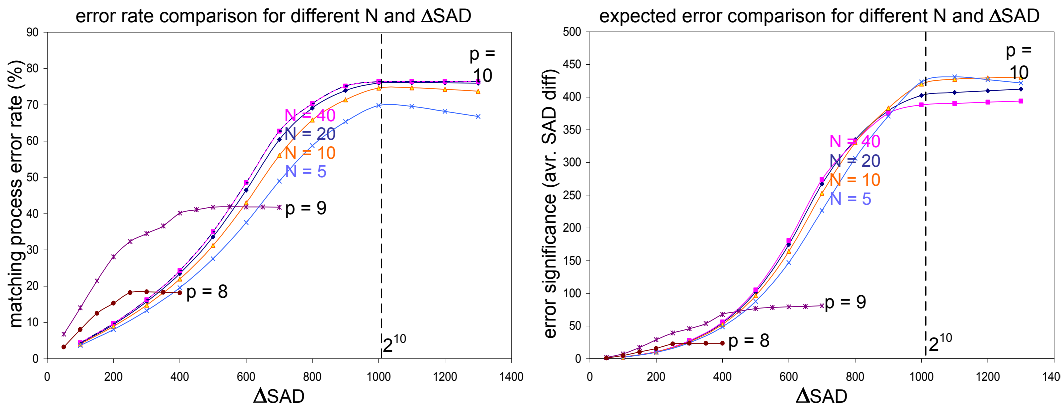


Figure 3.12: Effect of ΔSAD, N and fault location parameter p with respect to their impacts on the error rate P_E and expected error significance $\bar{E}$ based on our SA fault effect model (3.7),(3.8) and $P_{minSAD}(D)$ obtained from the CIF *foreman* sequence.

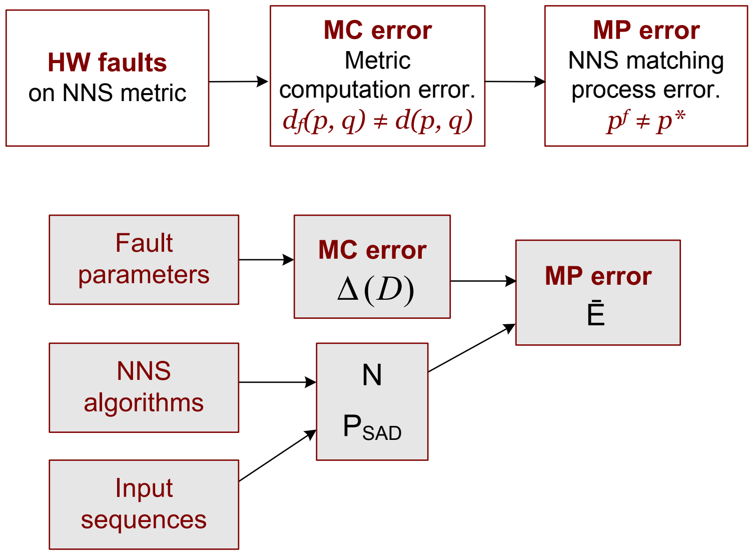


Figure 3.13: Illustrates that input sequence, fault parameters, and search algorithms all affect the matching process performance. Hardware architecture on the other hand affect the distribution of potential fault locations. Error tolerant search algorithm can reduce the impact of a given fault. Error tolerant MMC architecture includes smaller percentage of critical/unacceptable fault locations.

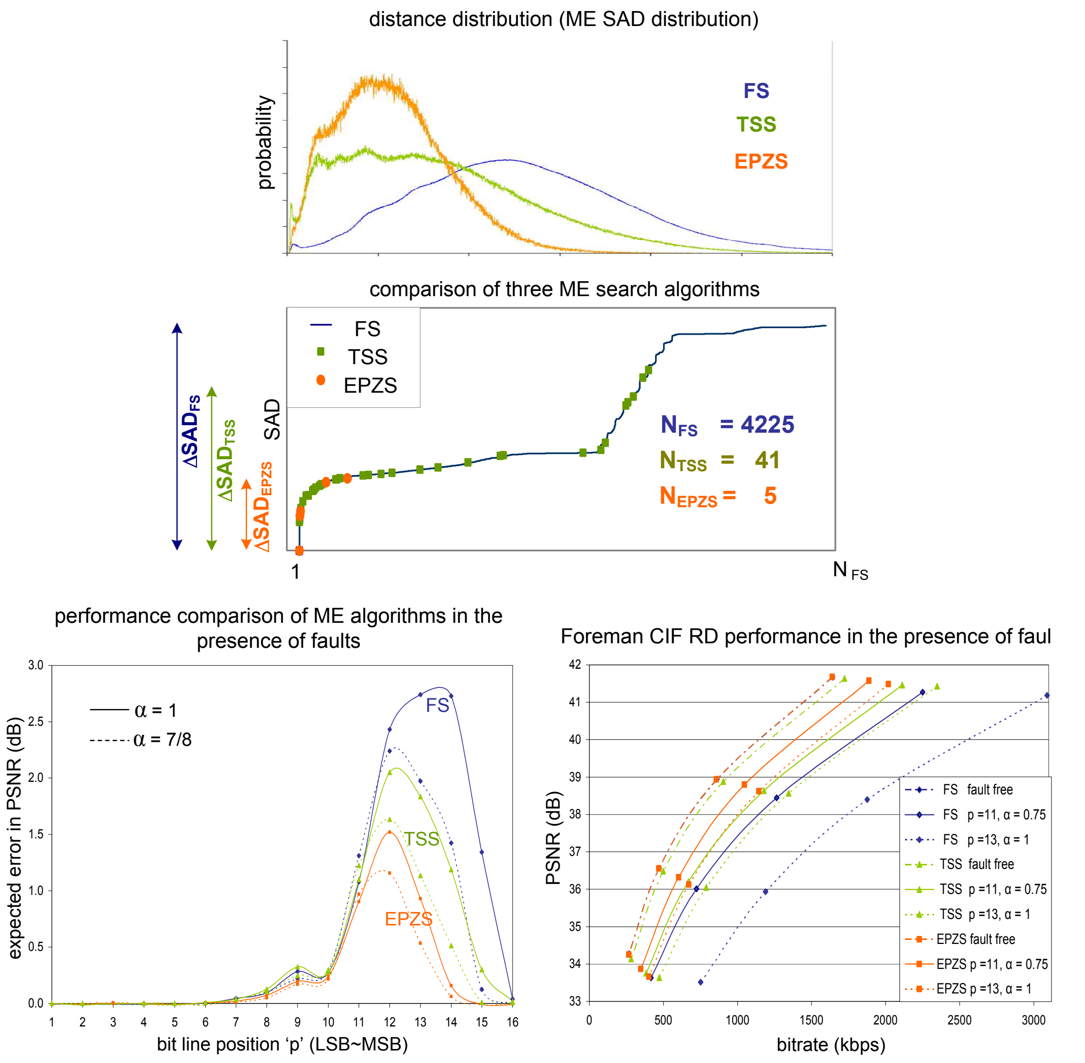


Figure 3.14: (top) comparison of three search algorithms FS, TSS, and EPZS with respect to ΔSAD and N. (left) ME algorithm comparison in coding efficiency loss for different SSA faults with parameters, $p = 0 \cdots 15$ from LSB to MSB at $\alpha = 1$ and 7/8. (right) RD impact of faults for different ME search algorithms.

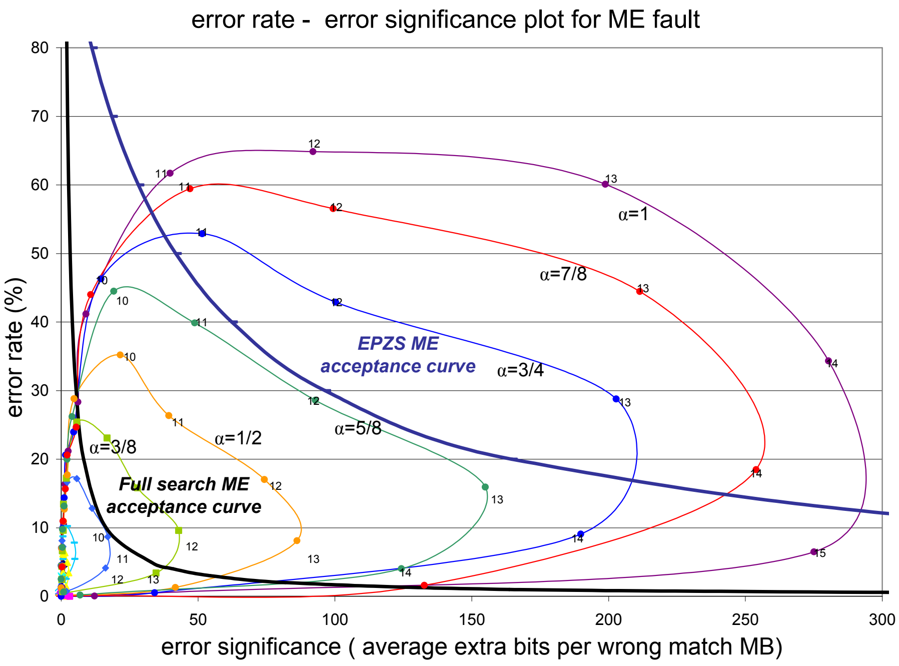


Figure 3.15: Illustration of different error tolerance level for different search algorithms. Impact of SA fault within the metric computation to search performance is measured with error rate and error significance.

3.5 Error Tolerant Algorithm Design Analysis for Nearest Neighbor Search

In Section 3.4, we introduced a function, $\Delta(D)$, to model the distortion in metric computation caused by a given set of multiple SA faults. Based on this model, we modeled the impact of faults on matching process by additionally considering the interaction between $\Delta(D)$ and the characteristics of a candidate set in terms of the number N and quality (distribution P_{SAD}). These attributes can be largely controlled by NNS search algorithm design while $\Delta(D)$ is determined solely by fault parameters i.e., locations, types, and dependencies. Therefore, in this section, based on our previous studies of MP error we define the characteristics of the search algorithm that lead to increased error tolerance and show how fault parameters and design choices made for the search process are related in terms of error tolerance. Note that our interest is not in accurate modeling of NNS search algorithms but in characterizing their average behavior with respect to our parameters of interest that are related to the error tolerance property.

The multiple faults effect model for the matching process in terms of P_E and $\bar{E}$ in (3.6), (3.7) shows that error robustness depends on both the number N and quality P_{SAD} of the candidates tested by NNS algorithm. Note that both P_E and $\bar{E}$ are a function of $\xi_N(d) = (1 - \hat{F}_{SAD}(\hat{d}))^N$, which is the only term associated with N. Since $\xi_N(d)$ decays exponentially as a function of N, for large enough N the difference in performance between different N values is negligible. Thus, in practice, for values of N encountered in most practical algorithms (e.g., $N > 10$) differences in overall error behavior are mostly determined by differences in quality of the candidates, rather than their number.

As to the quality of the candidates, P_{SAD} in our model, we approximate it by the range of P_{SAD} distribution (a coverage of SAD values) defined as $\Delta SAD = maxSAD - minSAD$ based on two reasons: i) Although it varies from block to block, our experiments show that average distribution given the same ΔSAD exhibits close to uniform distribution, and ii) *error region bounding interval* τ defined in (3.8) over which both P_E and $\bar{E}$ expectation computations are performed is bounded by ΔSAD. In other words, both P_E and $\bar{E}$ can be directly controlled by ΔSAD. Therefore, we approximate the quality of the candidates with one parameter ΔSAD in this section for the purpose of studying error tolerance property of NNS algorithms since it generally captures the main feature of P_{SAD} that is most related to the P_E and $\bar{E}$.

For given M multiple faults, there can be 2^M different τ values:

$$\tau_i \in \mathcal{T} = \{\tau | \tau = r - minR_S, r \in R_S\}$$

If $\Delta SAD > \tau_i \; \forall i$, both P_E and $\bar{E}$ are determined dominantly by $\Delta(D)$ function but not by ΔSAD. In other words, if a set of faults has all small p_i which decides R_S and consequently $\mathcal{T}$, the choice of NNS algorithm will not change the impact of faults and the impact itself of such faults is generally already insignificant due to their small p_i. Figure 3.14 (left) shows that three NNS algorithms for motion estimation application with different ΔSADs result in similar $\bar{E}$ for small p. On the other hand, if $\Delta SAD < \tau_i \; \exists i$, then $\tau_i = \Delta SAD \; \forall i \in \{ i \,|\, \tau_i > \Delta SAD\}$, resulting the *error region bounding interval* τ_i for P_E and $\bar{E}$ computation to be bounded by ΔSAD for some region of D. Therefore, the choice of NNS algorithm can influence the impact of faults considerably. If $\Delta SAD < \tau_i, \forall i$, all τ_i are completely bounded by ΔSAD and the error depends primarily on ΔSAD.

Figure 3.10 (right) provides a simple comparison of two candidate sets with different ΔSADs in the presence of a single SA fault and show how ΔSAD is linked with the expected error. Figure 3.12 illustrates our conclusions drawn from our model for a single SA fault case. It shows that P_E and $\bar{E}$ increase almost exponentially with ΔSAD and saturate when ΔSAD reaches $\tau = 2^p$ while the impact of N is minimal especially for large N.

Therefore, if our goal is to choose an NNS algorithm that reduces the impact of faults, we should choose one such that typical sets of MV candidates are as close as possible to optimal value (i.e., small ΔSAD). In practice, search algorithms satisfying this characteristic also show significant complexity reduction (small N) without having to compromise with the performance.

To evaluate the impact of NNS algorithm on error tolerance, three representative NNS search algorithms, full search (FS), three step search (TSS) [30], and enhanced predictive zonal search (EPZS) [14] are tested as they provide different combinations of N and ΔSAD parameters. FS exhaustively searches all candidates within the search window, and thus has the largest number of candidates N_{FS} and SAD range ΔSAD_{FS}. TSS successively evaluates sparsely distributed candidates and tries to follow the direction of minimum distortion to locate the smallest SAD. Although the number of candidates N_{TSS} is small, its SAD range ΔSAD_{TSS} remains relatively large. On the other hand, EPZS, a state of the art ME algorithm, considers a combination of optimized predictors and refinement process to locate the minimum distortion location. Unlike TSS, both the number of candidates N_{EPZS} and SAD range ΔSAD_{EPZS} tend to be quite small. As an example, when a search window of ± 32 is used, $N_{FS} = 4225$, $N_{TSS} = 41$, and $N_{EPZS} = 8.8$ on average for $Foreman$ CIF sequence. For the same sequence, we have $\Delta SAD_{FS} = 1.5 \times \Delta SAD_{TSS} = 9.75 \times \Delta SAD_{EPZS}$ on average. These properties are illustrated in Figure 3.14 (top), which shows the distribution of the SADs for all candidates of a given block, sorted by magnitude.

With these three algorithms, various sequences were tested with a series of fault parameters using a H.264 /MPEG-4 AVC baseline encoder. Only 16×16 block partitions, a single reference, and only integer-pel search were used for ME. Note that all experimental results presented from this point forward were performed under these same constraints. Figure 3.14 (left) provides the comparison of three algorithms in terms of PSNR degradation[3] due to a single SA fault with different parameters α and p while Figure 3.14 (right) depicts their RD performance. The

[3]BDPSNR (Bjontegaard Delta PSNR) [4] was used to calculate average PSNR difference between RD curves.

CIF resolution *Foreman* sequence was used for both graphs and other sequences tested showed similar results. Our experimental results are consistent with our earlier conclusions that the fault location parameters p and α mainly determines the fault impact on NNS performance while for a given fault location, NNS algorithm operating on smaller ΔSAD reduces the fault impact.

3.6 Error Tolerant Architecture Design Analysis

In addition to any error tolerance provided by a NNS algorithm, the different MMC architectures can also significantly influence the degree of error tolerance. For example, $\bar{E}$ can be reduced by more than 95% if a type-3 MMC architecture is used instead of type-1 (shown in Figure 3.2), when 16×16 block size is used for ME. In this section we show that MMC architecture with a perfectly balanced binary tree[4] structure (type-3) provides the highest error tolerance to SA faults.

As shown previously in Figure 3.2, there are several types of MMC architectures with different levels of parallelism, which can be grouped into either whole-block based (type-1,2,3) and sub-block based (type-4) structures. While whole-block based architectures allow more parallelized form, sub-block based ones reuse the architecture multiples times to perform whole block error computation. These two types of structures provide different error tolerance behavior and will be discussed separately. From here on, we omit the term "whole-block based" unless required for clarity.

Each MMC architecture can be uniquely represented as a binary tree with a given number of leaves $\mathcal{N}$, inner nodes $\mathcal{N} - 1$, and edges $2\mathcal{N} - 2$, where $\mathcal{N}$ corresponds to the number of pixels of the motion compensation block. Each edge

[4]a full binary tree in which all leaves are at depth n or n-1 for some n.

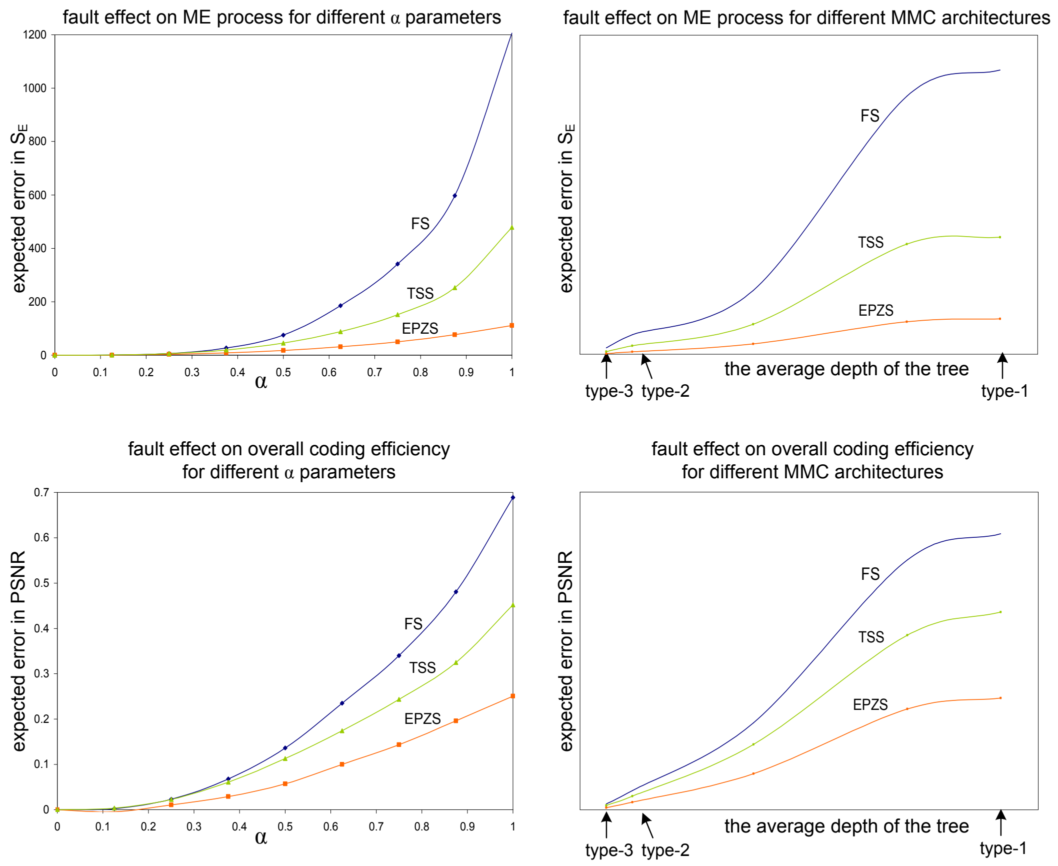


Figure 3.16: Average fault effect $\bar{E}$ on ME performance (top) and overall coding efficiency (bottom) are shown for different α parameters (left) and for different average binary tree depths ($D = \mathcal{N} \cdot E(\alpha)$) which describe different MMC architectures (right). Perfectly balanced trees (type-3 in Figure 3.2) show the minimum expected error introduced by a single SA fault.

corresponds to each data bus connecting adders in the MMC architecture and it consists of multiple bit lines. Each edge can be parameterized with α and we assume that every MMC architecture of our consideration uses the same number of bit lines for data buses which have the same α parameter. Then, every MMC architecture can be uniquely described by only the distribution of α parameter. (e.g., type-1 structure will give uniform distribution of α over all possible α value whereas type-3 will give high concentration for lower α and exponentially decreasing as α increases to 1).

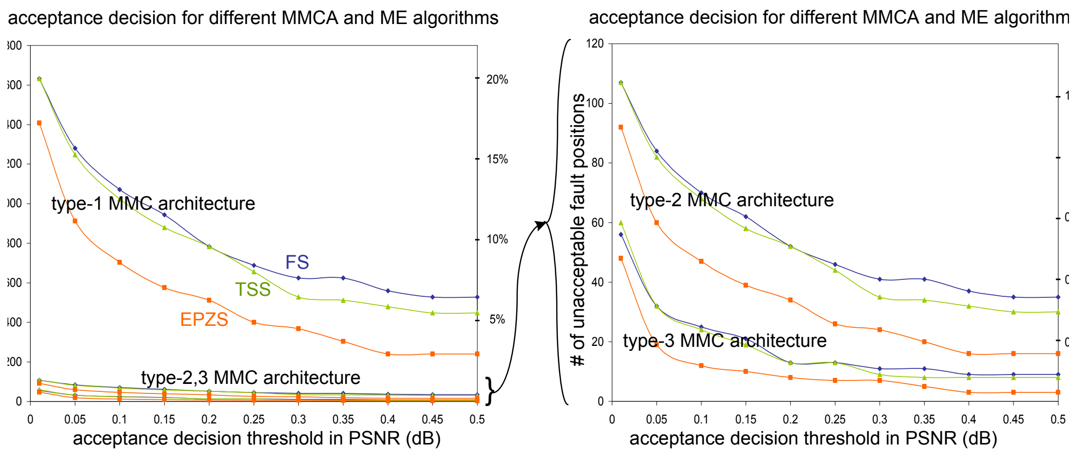


Figure 3.17: Experimental results on actual ET based decision example with single SA fault assumption in the context of H.264/AVC comparing different metric hardware architectures and search algorithms with respect to the percentage of unacceptable fault positions given different the decision thresholds. With a perfectly balanced tree MMC architecture (type-3), less than 1% of fault positions result in more than 0.01dB quality degradation.

Based on the fact that all MMC architectures have the same number of data buses and bit lines and the general assumption that random defects are distributed with equal probability across the metric computation circuit, the probability of fault occurrence is equal for all architectures. Therefore the expected value of additional error energy introduced by a single SA fault, $\mathbb{E}\left(\bar{E}|SSAF\right)$ indicates the error tolerance level of a given MMC architecture in the presence of a single SA fault. It can be represented as,

$$\mathbb{E}\left(\bar{E}|SSAF\right) = \sum_{\alpha_i \in A_k} \sum_{p_j \in P_{\alpha_i}} \bar{E}\left(\alpha_i, p_j\right) \cdot q = \sum_{\alpha_i \in A_k} \bar{E}_{\alpha_i} \cdot q = \sum_{\alpha} p\left(\alpha\right) \cdot \bar{E}_{\alpha}$$

where $\bar{E}_{\alpha_i} = \sum_{p_j \in P_{\alpha_i}} \bar{E}\left(\alpha_i, p_j\right)$ which is the same for all MMC architecture given α_i. A_k and P_{α_i} are the set of all data bus in MMC circuit k and of all bit lines belong to data bus α_i respectively. q is a probability of having a fault at certain position and it is equal probability over all fault locations. $p\left(\alpha\right)$ denotes the distribution of

α and it is determined by the MMC architecture. Based on our fault effect model studied in Section 3.4, $\bar{E}$ increases linearly with the fault parameter α with an assumption that both $P_{SAD}(D)$ and $P_{minSAD}(D)$ are uniform distributions over the entire range of D. Then,

$$\arg\min_{p_k(\alpha)} \mathbb{E}\left(\bar{E}|SSAF\right) = \arg\min_{p_k(\alpha)} \sum_{\alpha} p_k(\alpha) \cdot \bar{E}_{\alpha} = \arg\min_{p_k(\alpha)} D_k$$

where $D_k = \mathcal{N} \cdot \sum_{\alpha} p_k(\alpha) \cdot \alpha = \mathcal{N} \cdot E(\alpha)$ which is equivalent to the average depth of the binary tree that corresponds to the MMC architecture k. If $p_k(\alpha)$ determined by the MMC architecture k minimizes $\mathbb{E}\left(\bar{E}|SSAF\right)$, it also minimizes the average depth of the binary tree corresponding to that MMC architecture k. Therefore the MMC architecture corresponding to the binary tree with minimum average depth[5] (perfectly balanced binary tree) leads to the highest error tolerance and furthermore also maximize parallel computation. This conclusion holds more strongly when $\bar{E}$ increases with α superlinearly which is the case in practice as the above assumption is not generally true (Figure 3.16 (left)) mainly due to the $P_{minSAD}(D)$ distribution being more concentrated at the lower D such that minimum SAD becomes more likely to fall in the error region of $\Delta(D)$ function with higher α. More specifically, the relative reduction rate of $\bar{E}$ of the type-3 MMC structure compared to the others increases if the increasing rate of $\bar{E}$ vs. α is higher (e.g., exponential vs. linear increase) or if $\mathcal{N}$ increases[6]. Figure 3.16 illustrate how $\bar{E}$ increases with α (left) and with average binary tree depth describing different MMC architecture (right) for different ME algorithms in actual simulation.

[5]sketch of proof: a binary tree with $\mathcal{N}$ leaves in which the difference between maximum and minimum depths $D_{max} - D_{min}$ is greater than 1, can be reformed into the perfectly balanced tree by iteratively moving two leaves at D^i_{max} to D^i_{min} leaf. Each iteration i reduces average depth by $\left(D^i_{max} - D^i_{min} - 1\right)/\mathcal{N}$. Therefore perfectly balanced trees have the minimum average depth.

[6]for example, $\bar{E}$ of type-1, 2, and 3 increases in the order of $O\left(\mathcal{N}^2\right)$, $O(\mathcal{N})$, and $O(log_2\mathcal{N})$ respectively.

The same conclusion holds true for the multiple SA faults case although a formal argument establishing this property is somewhat involved, but its validity is essentially a consequence of the fact that $\bar{E}$ is an increasing function with each α_i of $\{f_i\}_{i=1}^{M}$.

Figure 3.17 shows how the percentage of unacceptable SSA fault locations given a MMC structure and search algorithm varies with the acceptance decision thresholds. It also provides comparison for three representative MMC architectures and search algorithms for the same process. This simulation result shows that even if acceptance decision threshold is as small as 0.01dB[7], still more than 99% of fault locations produce imperceptible degradation. In other words, more than 99% of defective metric computation circuits with SSA fault having type-3 architecture only produce less than 0.01dB degradation. This result confirms that the MMC architecture can affect the system level error tolerance property quite significantly.

Similarly for sub-block based MMC architectures which reuses its structure L times to perform one whole block error computation, the expected error for a SA fault f_i can be represented as

$$\bar{E}_{sub} = P_{sub}\left(f_i\right) \cdot \mathbb{E}_{sub}\left(S_E|f_i\right) = \frac{1}{L} P_{wh}\left(f_i\right) \cdot \left(\mathbb{E}_{wh}\left(S_E|f_i\right)\right)^L$$

Since a single SA fault in sub-block based structure has the same effect as that of L multiple independent SA faults with the same fault parameters in whole block based one, its impact increases by a factor of L. Therefore the fault impact for the sub-block based architectures increases exponentially with L, resulting in reduced error tolerance compared to the whole block based ones.

[7]In general, 0.1-0.2dB is considered to be an imperceptible quality difference in typical image/video coding applications.

3.7 Conclusions and Future Work

Based on the system-level error tolerance concept, we present a complete analysis of the effect of interconnect faults in NNS metric computation circuit. We provided a model to capture the effect of any fault (or combination of multiple faults) on the matching metric. We then describe how these errors in metric computation lead to errors in the matching process. Our motivation was twofold. First, we used this analysis to predict the behavior of NNS algorithms and MMC architectures in the presence of faults. Second, this analysis is a required step towards developing comprehensive testing techniques to identify acceptable error behavior. In particular, it will provide tools to tackle multiple fault scenarios, by discarding specific sets of faults (e.g., when one of the multiple faults is by itself unacceptable), establishing equivalence classes across faults (e.g., cases when dependent and independent faults may have the same impact), or determining how the parameters of the faults involved affect overall behavior (e.g., multiple faults with similar parameters may lead to relatively worse errors than sets of faults with differing characteristics).

Based on this model, we investigated the error tolerance behavior of nearest neighbor search (NNS) process in the presence of multiple hardware faults from both an algorithmic and a hardware architecture point of view by defining the characteristics of the search algorithm and hardware architecture that lead to increased error tolerance. More specifically, we investigate the relationship between fault locations and design choices made for the search process in terms of error tolerance and define the characteristics of the search algorithm that lead to increased error tolerance. We showed that error robustness depends on the number and quality of the candidates tested by NNS algorithm but the quality primarily influences ET level. We also showed that different hardware architectures performing the same metric computation can also significantly influence the degree of error tolerance and further showed that the optimal MMC hardware architecture in terms of error tolerance is perfectly balanced binary tree structures, which also allow the maximized parallel computing.

Motion estimation process for video coding is tested as an example application for our study to verify our models and results in actual practical application setting. Our simulation showed that search algorithms satisfying such characteristics (having a candidate set with smaller set size and having a distribution closer to nearest neighbors) also exhibit significant complexity reduction, apart from increased ET, without having to compromise with the performance. For example, in our experiments, enhanced predictive zonal search (EPZS) [14] algorithm which has these characteristics showed $0.01dB$ lower and up to $2.5dB$ higher performance than that of full search (FS) in fault-free and faulty cases, respectively, while reducing more than 99% complexity. Our simulation also showed that if optimal structure is used, the expected error due to a fault can be reduced by more than 95% and more than 99.2% of fault locations within matching metric computation circuits result in less than $0.01dB$ performance degradation.

Chapter 4

Conclusions and Research Directions

The subject of this thesis was studying similarity search problem within the scope of error tolerance concept. This thesis presented several methodologies to deal with the problems of system complexity and high vulnerability to hardware defects and fabrication process variability and consequently a lower yield rate.

Error Tolerance and Similarity Search

Error tolerance (ET) approach [6] is an exercise of designing and testing systems cost-effectively by exploiting the advantages of a controlled relaxation of system level output quality precision requirement. The basic theme of ET approach is to allow erroneous output but by inappreciable/imperceptible degree of system level quality degradation in order to simplify and optimize the circuit size and complexity, power consumption, costs as well as chip manufacturing yield rate. Motivation of ET approach is two-fold: By exploiting certain range of distortions/errors which lead to negligible impact on system level performance, i) a significant portion of manufactured dies with such minor imperfection of physical origin can be saved, thus increasing overall effective yield, and ii) considerable circuit simplification and high power efficiency is attainable by systematically and purposefully introducing such distortions/errors.

Considering the growing needs for dealing with a large often distributed volume of data and increasing mobile computing/telecommunication, complexity criterion (response time, circuit/power complexity) has become an important criterion for a successful design. Thus, the potential benefit of error tolerance concept in practical point of view can be enormous. Similarity search problem is one of the examples which could benefit from error tolerance approach due to its data and computation intensive characteristics. Also this is one of the best examples where approximate solution (relaxation from the exact solution requirement) is desirable and advantageous. In fact, almost all current research focus on proximity problems is to efficiently find the good approximate solutions.

Quantization Based Nearest-Neighbor-Preserving Metric Approximation Algorithm

Along with the improvement achieved from existing approaches for NNS problem in terms of the number of examined data during query process through database preprocessing techniques, our proposed approach introduces further significant complexity reduction by increasing the efficiency of metric computation. From the perspective of error tolerance (ET) concept specifically for the applications requiring similarity search process, we developed an efficient algorithm, called *query adaptive nearest-neighbor-preserving metric approximation algorithm*, which reduces computational burden of searching process by primarily focusing on the metric computation simplification. Proposed metric aims not at preserving measured distances but at preserving the fidelity of minimum distance ranking, while reducing potential wasting of resources in computing high precision metric for unlikely solutions/points. This metric approximation is not fixed for all potential queries but adaptively adjusted at each query process exploiting the information of query point to provide better performance complexity trade-off. Thus proposed method is intrinsically flexible from query to query and efficient with largely varying database /metric functions.

Our proposed approach employs quantization process within the metric, which is applied directly to data set points without having to compute actual distance to the query point. It entails nonlinear sensitivity to distance such that finer precision is maintained only where it is needed/important (the region of expected nearest neighbors) while unlikely regions to be nearest neighbors are very coarsely represented. Typically in our simulation with motion estimation, 1 bit quantizer per dimension/pixel is used and its resulting performance loss was negligible (on average 0.01dB loss). We provide an analytical formulation of the search performance measure, based on which the optimal quantizer is designed to maximize the fidelity of the minimum distance ranking.

Our proposed method is intrinsically more flexible and adaptable from query to query changes even with largely varying database or more dynamic environment (e.g., data streaming) or when there exist variation of metric functions and/or dimensionality change (e.g., user defined, controllable metric such as arbitrary selection of features/weightings), without having to rebuild the whole data structure or to perform transforms from scratch.

Our proposed approach can be also used for k-nearest neighbor search or orthogonal range search. To find the exact nearest-neighbor, this approach can be also used as a preliminary filtering step to filter out unlikely candidates and then refinement process can be performed within the remaining set. This method can be performed independently but also in parallel with most of existing preprocessing based algorithms.

More work needs to be done in terms of specific logic/circuit level design for efficient hardware implementation of quantizer. Additional future work is needed on more thorough analysis of performance factors such that given statistical information of certain application, one could accurately estimate how much complexity-performance trade-off can be expected. There is also a need to develop more concrete methodology in collecting statistical information from data when it is applied to certain index structure method. Also it would be interesting to see how this method can be incorporated with parallel and distributed index structures.

Furthermore, if possible, generalization of our proposed work can be developed to deal with more general metric function such as quadratic metric and to relax the constraint we posed in this thesis on metric structure (no cross-interference among dimensions in metric function).

The concept of our proposed approach can be extended to numerous problems involving search process such as combinations of queries, batch queries, classification problems, other proximity problems etc. But not limited to similarity search problem, the potential benefit of error tolerance concept can be reaped from variety of application areas.

Hardware Fault Effect Modeling for Nearest Neighbor Search Problem

Based on the system-level error tolerance concept, we present a complete analysis of the effect of interconnect faults in NNS metric computation circuit. We provided a model to capture the effect of any fault (or combination of multiple faults) on the matching metric. We then describe how these errors in metric computation lead to errors in the matching process. Our motivation was twofold. First, we used this analysis to predict the behavior of NNS algorithms and MMC architectures in the presence of faults. Second, this analysis is a required step towards developing comprehensive testing techniques to identify acceptable error behavior.

Based on this model, we investigated the error tolerance behavior of nearest neighbor search (NNS) process in the presence of multiple hardware faults from both an algorithmic and a hardware architecture point of view by defining the characteristics of the search algorithm and hardware architecture that lead to increased error tolerance. More specifically, we investigate the relationship between fault locations and design choices made for the search process in terms of error tolerance and define the characteristics of the search algorithm that lead to increased error tolerance. We showed that error robustness depends on the number and quality of the candidates tested by NNS algorithm but the quality primarily influences ET level. We also showed that different hardware architectures performing the same metric computation can also significantly influence the degree of error tolerance and further showed that the optimal MMC hardware architecture in terms of error tolerance is perfectly balanced binary tree structures, which also allow the maximized parallel computing.

Motion estimation process for video coding is tested as an example application for our study to verify our models and results in actual practical application setting. Our simulation showed that search algorithms satisfying such characteristics (having a candidate set with smaller set size and having a distribution closer to nearest neighbors) also exhibit significant complexity reduction, apart from increased ET, without having to compromise with the performance. For example, in our experiments, enhanced predictive zonal search (EPZS) [14] algorithm which has these characteristics showed $0.01dB$ lower and up to $2.5dB$ higher performance than that of full search (FS) in fault-free and faulty cases, respectively, while reducing more than 99% complexity. Our simulation also showed that if optimal structure is used, the expected error due to a fault can be reduced by more than 95% and more than 99.2% of fault locations within matching metric computation circuits result in less than $0.01dB$ performance degradation.

Based on the fault effect model we presented, we need to develop comprehensive testing techniques to identify acceptable error behavior. In particular, based on this model, we need to tackle multiple fault scenarios, by discarding specific sets of faults (e.g., when one of the multiple faults is by itself unacceptable), establishing equivalence classes across faults (e.g., cases when dependent and independent faults may have the same impact), or determining how the parameters of the faults involved affect overall behavior (e.g., multiple faults with similar parameters may lead to relatively worse errors than sets of faults with differing characteristics).

Bibliography

[1] http://biron.usc.edu/wiki/index.php/error_tolerant_computing.

[2] ITU (International Telecommunication Union) Study Group VQEG (The Video Quality Experts' Group), ftp://ftp.its.bldrdoc.gov/dist/ituvidq.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is 'nearest neighbor' meaningful? In *Lecture Notes in Computer Science*, volume 1540, page 217235, 1999.

[4] G. Bjontegaard. Calculation of average psnr differences between rd-curves. In *ITU-T Video Coding Experts Group, document VCEG-M33*, 2001.

[5] M. A. Breuer. Determining error rate in error tolerant VLSI chips. In *Proceedings of IEEE Int. Workshop on Electronic Design, Test and Applications DELTA04*, 2004.

[6] M. A. Breuer, S. K. Gupta, and T. M. Mak. Defect and error tolerance in the presence of massive numbers of defects. *IEEE Design & Test of Comp.*, 21:216–227, May–June 2004.

[7] T. K. Callaway and E. E. Swartzlander. Optimizing arithmetic elements for signal processing. In *VLSI Signal Processing, V*, pages 91–100, 1992.

[8] Y. Chan and S. Y. Kung. Multi-level pixel difference classification methods. In *in Proc. ICIP*, page 252255, 1995.

[9] H. Y. Cheong, I. Chong, and A. Ortega. Computation error tolerance in motion estimation algorithms. In *IEEE International Conference on Image Processing, ICIP'06*, Oct. 2006.

[10] H. Y. Cheong and A. Ortega. System level fault tolerant motion estimation algorithms & techniques. Technical report, SIPI, Univ. of Southern California, 2006.

[11] H. Y. Cheong and A. Ortega. Distance quantization method for fast nearest neighbor search computations with applications to motion estimation. In *Asilomar Conference on Signals, Systems and Computers*, Nov. 2007.

[12] H. Y. Cheong and A. Ortega. Motion estimation performance models with application to hardware error tolerance. In *Visual Communications and Image Processing VCIP'07*, Jan. 2007.

[13] H. Y. Cheong and A. Ortega. Quantization based nearest-neighbor-preserving metric approximation. In *IEEE International Conference on Image Processing, ICIP'08*, Oct. 2008.

[14] H. Y. Cheong and A. M. Tourapis. Fast motion estimation within the h.264 codec. In *Proc.IEEE Int. Conf. on MultiMedia and Expo (ICME-2003)*, July 2003.

[15] K. T. Choi, S. C. Chan, and T. S. Ng. A new fast motion estimation algorithm using hexagonal subsampling pattern and multiple candidates search. In *in Proc. ICIP. 2*, page 497500, 1996.

[16] I. Chong, H. Y. Cheong, and A. Ortega. New quality metrics for multimedia compression using faulty hardware. In *Proc. of International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, 2006.

[17] I. Chong and A. Ortega. Hardware testing for error tolerant multimedia compression based on linear transforms. In *Proc. of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT'05*, pages 523–534, 2005.

[18] H. Chung and A. Ortega. Analysis and testing for error tolerant motion estimation. In *Proc. of IEEE Int. Symp. on Defect Fault Tolerance in VLSI Syst.*, pages 514–522, 2005.

[19] P. Ciaccia, M. Patella, and P. Zezula. A cost model for similarity queries in metric spaces. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 59–68, 1998.

[20] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. In *IEEE Transactions on Information Theory*, volume 13, pages 21–27, 1967.

[21] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harsbman. Indexing by latent semantic analysis. In *Journal of the Society for Information Sciences*, volume 41, pages 391–407, 1990.

[22] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. In *Journal of the American Society for Information Science*, volume 41, 1990.

[23] L. Devroye and T. J. Wagner. *Nearest neighbor methods in discrimination.* Handbook of Statistics, 1982.

[24] A. Gersho and R. M. Gray. *Vector Quanfization and Signal Compression*. Kluwer Academic, 1991.

[25] Z. He, C. Tsui, K. Chan, and M. Liou. Low-power VLSI design for motion estimation using adaptive pixel truncation. In *IEEE Transaction on CSVT 10*, 2000.

[26] H. Hotelling. Analysis of a complex of statistical variables into principal components. In *Journal of Educational Psychology*, volume 27, pages 417–441, 1933.

[27] L. M. Huisman. Diagnosing arbitrary defects in logic designs using single location at a time (SLAT). *IEEE TCAD*, 23:91–101, 2004.

[28] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Symposium on Theory of computing*, 1998.

[29] R. Karri, K. Hogstedt, and A. Orailoglu. Computer-aided design of fault-tolerant VLSI systems. *IEEE Design & Test of Computers*, 13:88 – 96, 1996.

[30] T. Koga, K. Linuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion-compensated interframe coding for video conferencing. In *IEEE NTC*, pages 531–534, 1981.

[31] T. Kolda, R. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385–482, 2003.

[32] I. Koren and Z. Koren. Defect tolerant VLSI circuits: Techniques and yield analysis. In *Proceedings of the IEEE, Vol. 86*, pages 1817–1836, San Francisco, CA, Sept. 1998.

[33] I. Koren and C. M. Krishna. *Fault Tolerant Systems*. Morgan Kaufmann, United States, 2007.

[34] R. Leachman and C. Berglund. Systematic mechanisms-limited yield assessment survey. Competitive semiconductor manufacturing program, UC Berkeley, 2003.

[35] S. Lee, J. M. Kim, and S. I. Chae. New motion estimation algorithm using adaptively quantized low bit-resolution image and its VLSI architecture formpeg video encoding. In *IEEE Trans. CSVT. 8*, 1998.

[36] B. Liu and A. Zaccarin. New fast algorithm for motion estimation of block motion vectors. In *IEEE Trans. CSVT. 3*, page 148157, 1993.

[37] M. Love. Probability theory. In *Graduate Texts in Mathematics II, Springer-Verlag*, volume 46, 1978.

[38] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction. In *Proceedings of DARPA Broadcast News Workshop*, 1999.

[39] J. Marks, B. Andalman, P. Beardsley, W. Freeman, S. Gibson, J. Hodgins, and T. Kang. Design galleries: A general approach to setting parameters for computer graphics and animation. In *SIGGRAPH Conference Proceedings*, pages 389–400, 1997.

[40] B. Natarajan, V. Bhaskaran, and K. Konstantinides. Low-complexity block-based motion estimation via one-bit transforms. In *IEEE Trans. Circuits Syst. Video Technol. vol. 7*, page 702706, 1997.

[41] S. M. Omohundro. Efficient algorithms with neural network behaviour. In *Journal of Complex Systems*, volume 1, page 273347, 1987.

[42] P. Pirsch, N. Demassieux, and W. Gehrke. VLSI architectures for video compression-a survey. In *Proc. IEEE*, volume 83(2), pages 220–246, Feb. 1995.

[43] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice-Hall, 1995.

[44] A. A. Salamov and V. V. Solovyev. Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments. *Journal of Molecular Biology*, 247(1):11–15, 1995.

[45] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[46] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, MA, 1980.

[47] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, San Francisco, 2006.

[48] T. Sellis, N. Roussopoulos, and C. FaIoutsos. Multidimensional access methods: Trees have grown everywhere. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB)*, 1997.

[49] J. C. Spall. Introduction to stochastic search and optimization: Estimation, simulation, and control. *Wiley*, 2003.

[50] O. Stern and H. J. Wunderlich. Simulation results of an efficient defect analysis procedure. In *Proc. IEEE Int. Test Conf. on TEST: The Next 25 Years*, pages 729–738, Oct. 1994.

[51] V. Torczon. On the convergence of pattern search algorithms. In *SIAM Journal on Optimization*, volume 7, pages 1–25, 1997.

[52] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. In *Information Processing Letters*, volume 40, page 175179, 1991.

[53] P. Zezula, P. Savino, G. Amato, and F. Rabitti. Approximate similarity retrieval with m-trees. In *Very Large Databases Journal*, volume 7, pages 275–293, 1998.