

# CYBER FORENSICS: APPLICATION OF NORMALISED COMPRESSION DISTANCE FOR CROSS DRIVE CORRELATION

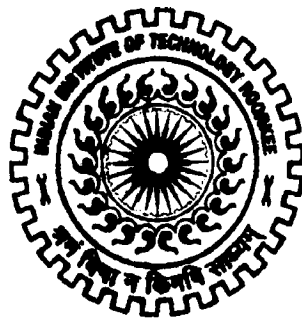
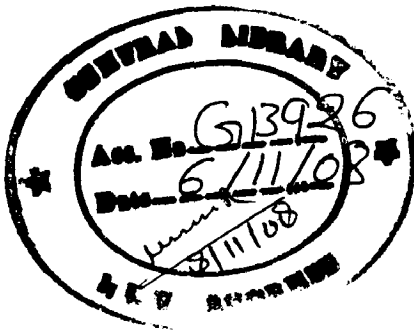
A DISSERTATION

*Submitted in partial fulfillment of the  
requirements for the award of the degree*

*of*  
**MASTER OF TECHNOLOGY**  
*in*  
**INFORMATION TECHNOLOGY**

By

**GUBBA RAMESH**



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE  
ROORKEE - 247 667 (INDIA)  
JUNE, 2008**

12

## CANDIDATE'S DECLARATION


---

I hereby declare that the work, which is being presented in the dissertation entitled "**CYBER FORENSICS: APPLICATION OF NORMALISED COMPRESSION DISTANCE FOR CROSS DRIVE CORRELATION**" towards the partial fulfillment of the requirement for the award of the degree of **Master of Technology in Information Technology** submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee (India) is an authentic record of my own work carried out during the period from June 2007 to June 2008, under the guidance of **Dr. R. C. Joshi, Professor, Department of Electronics and Computer Engineering, IIT Roorkee.**

I have not submitted the matter embodied in this dissertation for the award of any other degree or diploma.

Date: 17 Jun 08

Place: Roorkee



(GUBBA RAMESH)

---

## CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 18.6.08

Place: Roorkee



(Dr. R. C. JOSHI)

Professor

Department of Electronics and Computer Engineering

IIT Roorkee – 247 667

## ACKNOWLEDGEMENTS

---

I would like to take this opportunity to extend my heartfelt gratitude to my guide and mentor **Dr. R. C. Joshi**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, for his trust in my work, his able guidance, regular source of encouragement and assistance throughout this dissertation work. I would state that the dissertation work would not have been in the present shape without his inspirational support and I consider myself fortunate to have done my dissertation under him.

I also extend my sincere thanks to **Mr Raj**, 'Information Security' Lab in-charge, for providing facilities for the work. I also wish to thank Ms. Anjali Sardana and my co-students for their valuable suggestions and timely help whenever asked for.

I would also like to express my gratitude to my parents for everything that they have done for me, to my wife for putting up to my tantrums but still supporting and encouraging me and finally to my son Abhishek who sacrificed his quota of playtime with me. All of this would have been impossible without their constant support.

**GUBBA RAMESH**

## Abstract

---

In this work a new approach of automated Cross-Drive correlation, in computer forensics, is presented. This approach uses the concept of Normalized Information Distance(NID) that helps to derive drive similarity correlation between a pair of disk images. The algorithm uses the Normalized Compression Distance (NCD) which is the implementation approximation of NID. The method proposed is a parameter free correlation unlike the previous work which is based on generation of common features as parameters of comparison and correlation. The ever increasing capacities of digital storage devices and their rapid proliferation makes parameter based systems more time consuming as the generation of features or parameters would take a considerable amount of time. However, parameter free algorithm would provide quick and more complete leads and clues to the investigator so that he can focus only on the highlighted subset of input datasets for further detailed investigation. The main advantages of NCD based cross drive correlation are: examination of data for generating forensic features as parameters is not required, savings on time and resources that otherwise would be required for forensic features extraction, deep knowledge of the underlying data is not required, it would detect all similarities simultaneously, it would automatically select dominant shared features in all pairwise comparisons and can be used effectively for heterogeneous data. The algorithm works in three main stages: conversion of the acquired image to a reduced signature, NCD correlation and finally calculation of pairwise correlation score with graphical representation. Experiments on disk images of 200MB were conducted and the programs developed, without many modifications, can be easily scaled to inputs of sizes in Giga Bytes.

# CONTENTS

---

<b>Candidate's Declaration and Certificate.....</b>	<b>i</b>
<b>Acknowledgements.....</b>	<b>ii</b>
<b>Abstract.....</b>	<b>iii</b>
<b>Table of Contents.....</b>	<b>iv</b>
<b>CHAPTER 1</b>	<b>Introduction and Statement of the Problem.....1</b>
1.1	Introduction.....1
1.2	Motivation.....2
1.3	Problem Statement.....3
1.4	Organization of the Report.....4
<b>CHAPTER 2</b>	<b>Background and Literature Review.....6</b>
2.1	Cyber Forensic Components.....6
2.2	Investigating Evidence Spanning Multiple Disks.....7
2.3	Existing Tools and Research Gaps.....7
2.4	Cross Drive Correlation.....9
<b>CHAPTER 3</b>	<b>Framework for NCD Similarity based Correlation.....12</b>
3.1	Framework Overview and Sub-Tasks.....12
3.1.1	Disk Image Preprocessing.....14
3.1.2	NCD Similarity Correlation.....14
3.1.3	Reports and Graphical Output.....15
3.1.4	Data Blocks Extraction.....15

<b>CHAPTER 4</b>	<b>Techniques Used in Framework.....</b>	<b>16</b>
4.1	Normalised Compression Distance.....	16
4.2	Byte based File Statistics.....	19
<b>CHAPTER 5</b>	<b>Framework Implementation.....</b>	<b>20</b>
5.1	System Requirements.....	20
5.2	Implementation of Disk Image Preprocessing Module.....	20
5.3	Implementation of NCD Correlation Module.....	21
5.4	Outputs and Data Blocks Extraction .....	24
5.5	Generation of Test Images.....	26
<b>CHAPTER 6</b>	<b>Results and Discussion.....</b>	<b>27</b>
6.1	Accuracy and Speed of Similarity Detection.....	27
6.1.1	Window Size of Data Reduction.....	27
6.1.2	Window Size for Similarity Comparison.....	28
6.1.3	Compression Algorithm.....	28
6.1.4	Random Noise Resistance of NCD.....	29
6.1.5	Effect of Relative Offsets of Data.....	30
6.2	Optimization of Graphical Display.....	31
6.2.1	Threshold Values of NCD.....	32
6.2.2	Noise Elimination during Data Reduction.....	34
6.3	Limitations.....	36
6.4	Validation of Test Results.....	37

<b>CHAPTER 7</b>	<b>Conclusion and Future Work.....</b>	<b>39</b>
7.1	Conclusion.....	39
7.2	Scope for Future Work.....	39
<b>REFERENCES.....</b>		<b>41</b>
<b>PUBLICATIONS.....</b>		<b>43</b>
<b>APPENDICES</b>		
A	Source Code.....	I
B	Program GUI Screenshot.....	XXIX

**1.1 Introduction**

In this era every aspect of our life is touched by computers and other digital devices. We use them for shopping, business, communication etc. Any computer/digital device can be used for multiple purposes. The pervasiveness of these devices has increasingly linked them to crimes and incidents. As a consequence almost all criminal activities might leave behind some sort of digital trace. Many crimes normally not thought of as cyber crimes are requiring cyber forensics. Therefore, to initiate a criminal or civil prosecution we need to use scientifically derived and proven strategies of, 'Digital Forensic Investigations' or 'Cyber Forensics' that depend on sound forensic principles. The strategies for proving or disproving the crime scene hypothesis must be time-bound, efficient and accurate [1 and 2].

The methods and tools available to a digital forensic investigator today are basically single disk drive or disk image analyzers and perform the tasks of data carving, hash generation and analysis, e-mail header analysis etc. A typical crime scene today may consist of many digital storage devices of various shapes, capacity and functionality including those which are not present at the crime scene itself. A world wide terrorist network is a very good example to understand. The evidence needs to be culled out or interpreted by analyzing together all the seized potential digital evidence sources like, personal computers, email and chat accounts, ISP records, mobile phones, answering machines etc. These evidence sources might be many and from different geographical locations. The primary effort would be to see if all the digital devices seized have some underlying relations that can throw light on the investigations. Towards this, Cross-Drive Analysis (CDA) [3] is a concept of cross drive correlation proposed for investigating evidence spanning multiple drives by generating pseudo-unique forensic features like credit card numbers.



In this dissertation another approach of cross drive correlation is presented. In this new approach the similarity metric 'Normalized Compression Distance' (NCD) [4] has been applied to get a correlation between a pair of disk drives by using the raw data. This method would greatly reduce the load of an investigator as quick and accurate leads can be gathered without actually parsing and understanding the data to generate forensic features. This new method will not be a total substitute but an alternate approach for faster results. Detailed investigations and the CDA of [3] can be used subsequently on minimized input datasets.

## 1.2 Motivation

Finding evidence on a hard disk is like finding a needle in a haystack. Laptops and desktops with 1TB of storage, mobile phones with tens of GB of storage are already in market. This large volume of data makes the task of investigation more complex and slow. In case of large number of drive images pertaining to a single case, the investigator would want to identify hot drives and the relations between the drives with reasonable accuracy and in minimum time. The present practice is to carve valid file objects and manually assess them for information for each drive image. The individual results are then correlated by manual assessments. This practice may not account for all underlying relations or correlations unless the investigator has a keen eye for details.

The enhancement to this manual practice is the use of Forensic Feature Extraction (FFE) and Cross-Drive Analysis (CDA) proposed in [3] which helps in analyzing large data sets of disk images for highlighting correlations in an automated manner. FFE-CDA uses statistical techniques for analyzing information within a single disk image and across multiple disk images based on extracted pseudo-unique identifier as features like social security numbers and credit card numbers. However, the problems with this approach are:

- Low level examination of each individual drive is required to generate forensic features.
- Either the drives need to be mountable i.e the meta data has to be intact or file carving techniques need to be used which can be time consuming with many false positives.
- Information from deleted and slack space has to be retrieved to generate forensic features.
- Some relevant features may be missed out.
- Detailed and deep domain knowledge of data in disk images is required. For example[3] in case of credit card number feature extractor the following needs to be ensured:
  - String of 14-16 digits, no spaces or other characters
  - No single digit is repeated more than 7 times
  - Pairs repeated not more than 5 times
  - Format validity; first 4 digits denotes the card issuer and length of string is consistent with the issuer
  - Sequence of digits as per validation algorithm

These observations emphasize the need for a more completely automatic tool that provides cross drive correlation at the bottommost physical data level(sectors on a hard disk) by just using the raw data on the disk, without bothering about the specific type and nature of the data which is otherwise essential for feature extraction.

### **1.3 Problem Statement**

The requirement is to provide Cross Drive Correlation of the drive images without actually understanding or parsing the raw bytes and just by using the data signatures of the raw data. The problem can be divided into following:

- Devise a method to compare the raw data of one image with another, irrespective of the type of file system, irrespective of operating system, irrespective of data pertaining to existing or deleted file; while taking into account the data residing in hidden partitions, volume slack, file slack and masquerading file types.
- Achieve computational efficiency and reasonable accuracy of results.
- Devise graphical display representation and correlation score formulation.
- Extract data that satisfies the similarity threshold.
- Achieve a preliminary analysis of evidence spanning multiple disks, by avoiding the cumbersome and time consuming Examination phase of the Forensic process.

#### **1.4 Organization of the Report**

The complete work on use of Normalized Compression Distance (NCD) for cross drive correlation is presented in this report in the following format:

Chapter 2 contains the background and literature review. In this chapter the Digital Forensic Framework or process is stated and the existing techniques and the proposed technique are discussed in terms of the forensic process for investigating multiple drives. The existing research gaps are also highlighted.

Chapter 3 explains the NCD similarity correlation algorithm and the functions of various sub-tasks in it.

Chapter 4 explains the mathematical foundations of the techniques of Normalized Compression Distance as a similarity measure and Byte statistics for data reduction. These techniques are the foundation of the algorithm devised in this work.

Chapter 5 explains the implementation details and issues of the proposed strategy. The generation of test input data sets is also discussed.

Chapter 6 discusses the results. The various optimization issues and limitations are spelt out.

Chapter 7 concludes by summarizing the work and discussing its applicability. Areas where further work needs to be done are also listed out in this chapter.

## 2.1 Cyber Forensic Components

The Cyber or Digital Forensic Process as defined by Digital Forensic Research Workshop (DFRWS) in [5] and [6] has the following components:

- (i) **Collection.** Data related to a specific event is identified, labeled, recorded, and collected, and its integrity is preserved. Here media is transformed into data.
- (ii) **Examination.** Identification and extraction of relevant information from the collected data while protecting its integrity using a combination of automated tools and manual processes. Here data is transformed into information.
- (iii) **Analysis.** This involves analyzing the results of the examination to derive useful information that helps the investigation. Here information is transformed into evidence.
- (iv) **Reporting.** Reporting the results of the analysis, describing the actions performed, determining what other actions need to be performed, and recommending improvements to policies, guidelines, procedures, tools, and other aspects. Generated evidence is used to formulate reports, prepare charts and support decisions.

Fig 2.1 depicts the framework in terms of activities and the inputs and outputs associated with each of the activity.

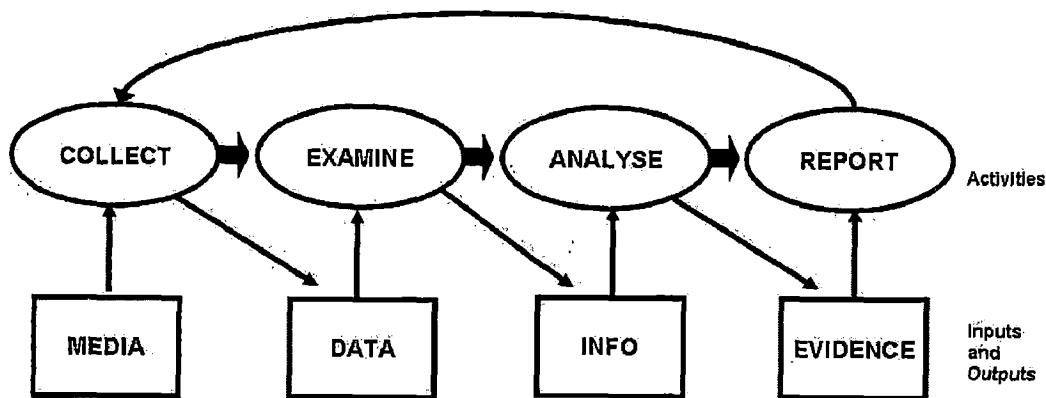


Fig 2.1: Components of DFRWS Digital Forensic Process

## 2.2 Investigating Evidence Spanning Multiple Disks

The requirement can best be explained by an example. In a case of investigating a terrorist network, digital storage media of many individuals, many organizations separated geographically may be seized as potential source of evidence and intelligence. Here the investigation, as per the existing norms, would iteratively focus on 'examination' and 'analysis' of each piece of digital media. The examination would typically consist of data carving, key word search, hash verification etc. The correlation between the various data sources would be established, if any, by manually perusing the individual reports.

## 2.3 Existing Tools and Research Gaps

The existing open and commercial tools do not provide any support for automated analysis for correlation between two or more disk images, in case of evidence spanning multiple devices. The tools primarily perform data carving and examination on a *single disk image*. Some of these are discussed below.

EnCase is the most popular [7] computer investigation software. It supports many types of file systems. It lists files, directories and recovers deleted and slack files. It performs keyword searches, hash analysis of known files and duplicates and generates timelines of file activity, etc. The software has its own scripting language support for additional customization.

Forensic Tool Kit(FTK) by AccessData which is the next most popular tool is revered for its searching capabilities and application level analysis like analysis of e-mail headers. It also performs other single disk activities as in case of Encase. The hash based comparison feature tells if the two files are same or different.

WinHex, by X-Ways software technologies AG, which basically is a advanced hex editor, provides a feature of position based byte by byte comparison and reports the total number of differences or similarities. This information though would indicate how different the two images are but would not show any other details.

Other some what less popular tools are ProDiscover, SleuthKit, Autopsy etc. All these perform more or less the same type of functions and there is no evidence of any feature specifically catered for analyzing multiple drives. Hence, there is a need to carry out research for accurate and fast correlation between multiple disk images so that quick preliminary clues and leads can be gathered in any investigation. The research gaps in this respect can be summarized as:

- Techniques for correlating many disks each of capacity in Giga Bytes or Tera Bytes.
- Tools to automate the correlation techniques efficiently.
- Visualization and reporting of such analysis for faster interpretation.

## 2.4 Cross Drive Correlation

Traditionally each piece of evidence is subjected to the various phases of the forensic process as shown in Fig 2.1. The sub-tasks in the 'Examination' and 'Analysis' phase (Examination: Traceability, Pattern match, Hidden data discovery, etc; Analysis: Timelining, Link analysis, Spatial analysis, Traceability etc) would need to be applied to each individual item in the evidence bag. At the end the results would require comparison and manual inspection so that correlations are highlighted. This is a time consuming process and may not always lead to the desired results.

Simson L. Garfinkel has spelled out an architecture which uses the techniques of Forensic Feature Extraction (FFE) and Cross-Drive Analysis (CDA) in [3]. This architecture can be used to analyse large data sets of forensic disk images. It contains the following five tasks as shown in the Fig 2.2.

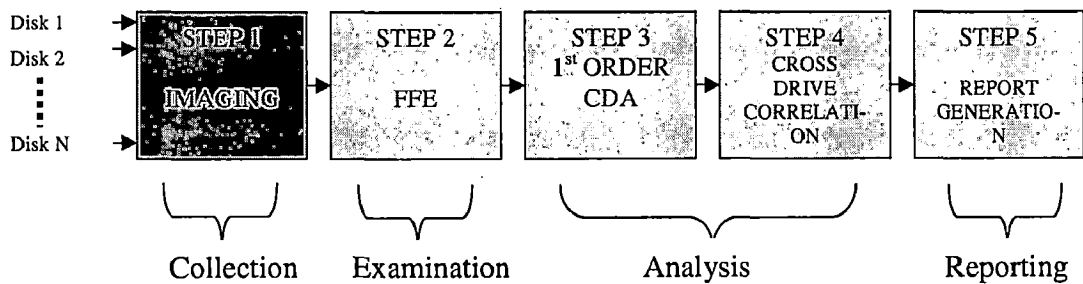


Fig 2.2: Forensic Feature Extraction (FFE) based Cross-Drive Analysis (CDA) Architecture-as Mapped to Digital Forensic Process.





$$DC(f) = \sum_{n=0}^D FP(f; dn) = \text{set of drives with feature } f \dots\dots\dots(2.4.2)$$

$$S2(d1; d2) = \sum_{n=0}^F FP(fn; d1)FP(fn; d2)/ DC (fn) \dots\dots\dots(2.4.3)$$

- (c) If features that are present in high concentrates on drives d1 and/or d2 should increase the weight so as to increase the score between a computer user who had exchanged emails with a known terrorist when compared with an individual who has only exchanged one or two emails with the terrorist then a scoring function, S3, can be defined as:

FC(f ; d) = count of feature f on drive d; then

$$S3(d1; d2) = \sum_{n=0}^F FC(fn; d1)FC(fn; d2)/ DC (fn) \dots\dots\dots(2.4.4)$$

Using these scoring functions several examples of single drive analysis, hot drive identification and social network discovery were presented in [3].

### 3.1 Framework Overview and Sub-Tasks

Unlike the FFE based CDA the NCD similarity based correlation process does not require the 'Examination' phase of the Forensic process as there is no requirement to parse and interpret the raw data beforehand. This would save precious amount of time as we can straight away zero-in on the suspected drives just by correlating the drives by using NCD similarity. However, if required, the extracted data blocks that meet the similarity constraints can be further subjected to FFE based CDA or some other method for a more thorough investigation. In the normal case the similarity based analysis would be the first pass and the FFE-CDA would be the optional second pass.

We assume that the similarities of our interest are not very minuscule, thereby having a good chance of being detected. It is also assumed that the operating system files and other common files have been removed during imaging in the 'Collection' phase and the data has been converted into a common format from various representation formats like ASCII, Unicode, big-endian, small-endian etc.

Figure 3.1 shows the Normalised Compression Distance based similarity correlation scheme in relation to the DFRWS forensic framework. The examination block is SKIPPED because features need not be generated.

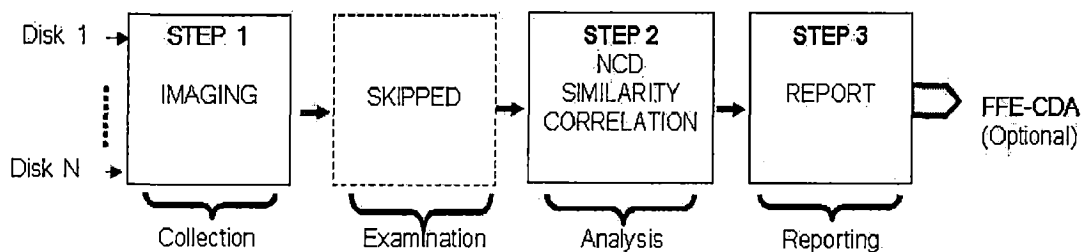


Fig 3.1: NCD Similarity Based Correlation Analysis

The proposed algorithm and the steps therein are explained below. The main blocks of the algorithm are: '*Disk Image Preprocessing*', '*NCD similarity Correlation*', '*Reports and Visualization*' and '*Data Block Extraction*'. The algorithm is depicted in Fig 3.2

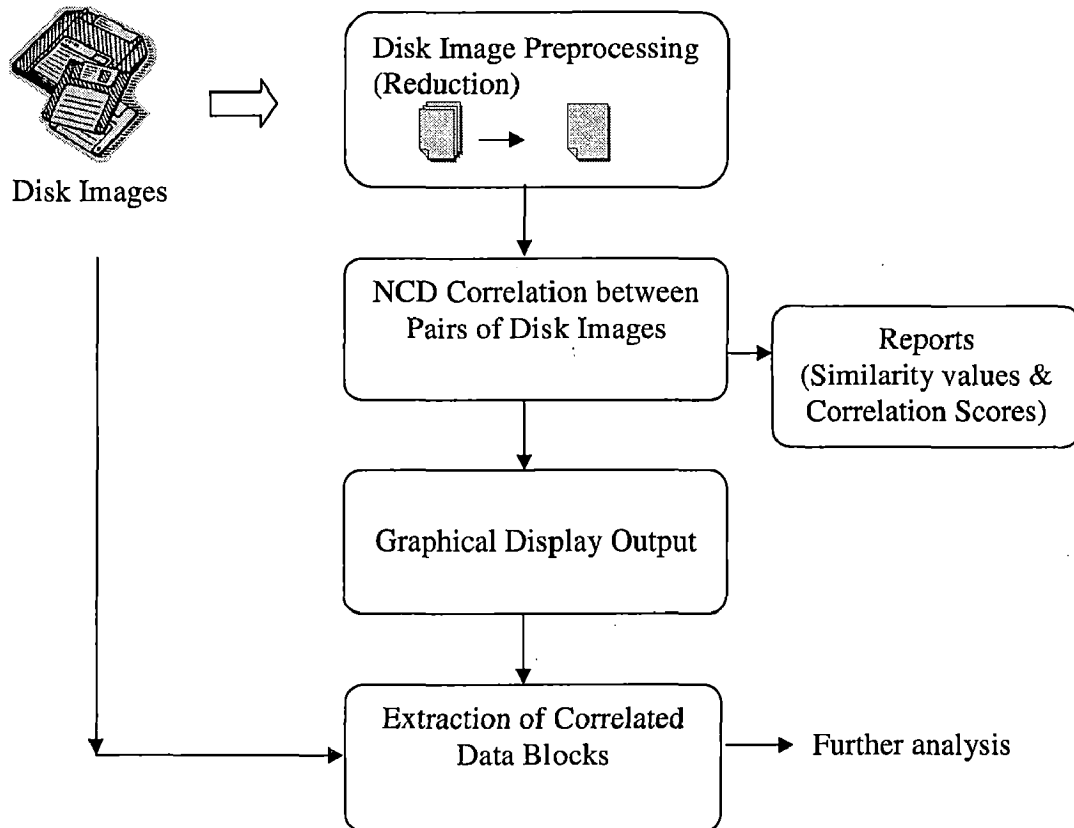


Fig 3.2 : Algorithm of NCD Based Disk Images Correlation

The general functionality of each sub-task in the architecture is discussed in subsequent paragraphs. The actual implementation details are mentioned in chapter 5 and chapter 6.

### **3.1.1 Disk Image Preprocessing**

The capacities of digital media can be in Giga Bytes or Tera Bytes. Correlating raw data of such large capacity disks, if possible, would consume tremendous amount of computational efforts and would defeat the aim of a quick investigation. It was seen that without preprocessing the activity of correlating two 200MB disk images took a minimum of 8 hours of computation on a normal desktop computer using 350KB NCD comparison window. Therefore a preprocessing block was introduced so that the image is reduced using file byte statistics [8] to produce a data signature. The same reduction if applied to all images does not alter the characteristics of the image in relation to each other and hence the reduced images can be correlated. This brings down the computational effort and the memory requirements. Images of size 200MB were reduced to about 400KB( 99.8 % reduction) and the computational effort came down within the range of 3 to 4 minutes.

### **3.1.2 NCD Similarity Correlation**

This module is the heart of the framework. Here information distance based similarity metric, Normalised Compression Distance, is used to detect all possible pairwise dominant similarities between the input disk images. Normalised Compression Distance is applied on the reduced input disk images. The correlation is block per block i.e the block size is the comparison window size. The comparison window is a sliding window which moves one window size at a time. The correlation score is calculated and reported based on certain fixed thresholds. The correlation information is depicted as a graph in accordance to the thresholds. This module needs to be optimized for fast and accurate results.

### **3.1.3 Reports and Graphical Output**

After completion of correlation between the pair of disk images, the results are reported in a file. The similarity correlation denoting the similarity values against the data block numbers is difficult to understand. Hence a graphical output provides a simple and effective visualization of the similarities between the two correlated images.

### **3.1.3 Data Blocks Extraction**

This is the final task wherein the similarity correlated data blocks, as per the threshold values, of the disk images being compared are extracted and saved as separate files. Further detailed investigation can be performed on these extracted files and these additional activities can be data carving, FFE based CDA or another iteration of NCD similarity correlation with smaller comparison window size for more accurate results.

---

---

#### 4.1 Normalised Compression Distance

Similarity [9] is a degree of likeness between two objects. So a similarity measure is a distance between the two objects being compared. There are many metrics available to express similarities like Cosine distance, Euclidean distance etc. These metrics require additional details as dimensions for arriving at the calculations. Moreover, some of these metrics provide absolute results meaning that the comparisons are not normalized.

The other class of similarity metric is Normalised Compression Distance(NCD) based on Normalised Information Distance (NID) which in turn is based on kolmogorov complexity. The mathematical definitions and explanations as in [4] and [9] are given below.

*Metric:* First of all let us consider when a distance can be termed a metric. A distance is a function  $D$  with nonnegative real values, defined on the Cartesian product  $X \times X$  of a set  $X$ . It is called a metric on  $X$  if for every  $x, y, z$  in  $X$ :

- $D(x, y) = 0$  iff  $x = y$  (the identity axiom).
- $D(x, y) + D(y, z) \geq D(x, z)$  (the triangle inequality).
- $D(x, y) = D(y, x)$  (the symmetry axiom).

A set  $X$  provided with a metric is called a metric space. For example, every set  $X$  has the trivial discrete metric  $D(x, y) = 0$  if  $x = y$  and  $D(x, y) = 1$  otherwise.

*Kolmogorov Complexity:* If  $x$  is a string then  $K(x)$  is the kolmogorov complexity of  $x$  and essentially is the shortest program which can generate  $x$ . Therefore the upper

bound is  $K(x)=|x|$ . As kolmogorov complexity is noncomputable, it is approximated using compression. If  $C^*$  and  $D^*$  are complimentary compression and decompression program lengths and  $C(x)$  is compressed length of  $x$  then

$$K(x) = C(x) + D^* \dots\dots\dots(4.1.1)$$

The conditional kolmogorov complexity  $K(x|y)$  of  $x$  relative to  $y$  defines the length of shortest program to generate  $x$  if  $y$  is given.  $K(x,y)$  is the length of shortest binary program to produce  $x$  and  $y$  and distinguish between them. It has been shown that there exists a constant  $c \geq 0$  and independent of  $x,y$  such that

$$K(x,y) = K(x) + K(y|K(x)) = K(y) + K(x|K(y)) \dots\dots\dots(4.1.2)$$

where the equalities hold up to 'c' additive precision [9].

*Normalised Information Distance:* Information distance is the length  $E(x,y)$  of a smallest program that can generate  $x$  from  $y$  and vice-versa. It has been stated as

$$E(x,y) = \max \{K(y|x),K(x|y)\} \dots\dots\dots(4.1.3)$$

upto an additive logarithmic term. Therefore,  $E(x,y)$  is a metric upto an additive logarithmic term. Further,  $E(x,y)$  is absolute and not relative or normalized. The normalized  $E(x,y)$  is termed Normalised Information Distance and defined as

$$NID(x,y) = \max \{K(y|x),K(x|y)\} / \max \{K(x),K(y)\} \dots\dots\dots(4.1.4)$$

$NID(x,y)$  is also noncomputable as it depends on kolmogorov complexity. Therefore,  $NID$  is approximated using a real world compressor that is normal and the metric is termed Normalised Compression Distance (NCD). A normal compressor has the following properties upto a logarithmic additive term.

- Idempotency:  $C(xx)=C(x)$  and  $C(y)=0$  if  $y=0$
- Monotonicity:  $C(xy) > C(x)$



- Symmetry:  $C(xy) = C(yx)$
- Distributivity:  $C(xy) + C(z) \leq C(xz) + C(yz)$

Therefore, NCD is also a metric upto an logarithmic additive term. Normalised Compression Distance has been stated in [4] and [9] as

$$NCD(x,y) = \frac{(C(xy) - \min\{C(x), C(y)\})}{\max\{C(x), C(y)\}} \dots\dots\dots(4.1.5)$$

The essential features of NCD similarity metric are:

- It is parameter free which means detailed domain knowledge and subject specific features are not essential.
- It is universal in the sense that it approximates the abstract similarity parameter based on the dominant features in all pairwise comparisons.
- The objects being compared can be from different realms.
- It is a general metric by the virtue of its applicability for varied data like text, music, sourcecode, executables, genomes etc.
- It captures every effective distance between the objects of comparison.
- The results are normalized and usually take values in the range of  $[0,1.0]$  . This means that the results are relative and not in absolute terms, making interpretation of the results very easy.
- As real compressors are space and time efficient, in certain applications they can more efficient than parameter based methods by an order of magnitude of 3 to 4 as stated in [10].
- NCD is random noise resistant to a large extent as shown in [11].
- Data to be compared need not be in a particular format.
- The objects being compared need not be of same dimensionality.

In the scheme of NCD similarity disk image correlation all data whether visible, hidden, fragmented or as part of file/partition/volume slack are automatically taken care of in the process of comparison.

## 4.2 Byte based File Statistics

Byte value file statistics have been used very effectively for identification and localization of data types within large scale file systems mainly as part of steganalysis in [8]. Thirteen statistics were assessed to determine the signature of a file type and of these *average, kurtosis, distribution of averages, standard deviation and distribution of standard deviation* were found to be adept at differentiating the data types.

However, the statistic selected has to be used in conjunction with a sliding window of appropriate size. In [8] window sizes in the range [256,1024] bytes were found to be optimum for the purpose of localization of files in a data set.

In this framework of NCD similarity correlation *byte average* statistic has been chosen to preprocess the data and reduce size while maintaining the individual characteristics of the objects being compared. Reduction window size of 512 bytes was used to achieve a fair amount of originality while attaining a good percentage of reduction.

### **5.1 System Requirements**

The hardware used was a standard desktop with a Pentium dual core and 1 GB RAM. The operating system on the desktop was Windows XP. The programs were developed in Delphi 7. There were no special resource or tweaking used because more than the performance aspect it was the validation of the concept that was the primary focus of the experiments as part of this work. The choice of the programming language was influenced by my previous familiarity and need for GUI for depicting results as charts for better visualization.

### **5.2 Implementation of Disk Image Preprocessing Module**

Disk image preprocessing is necessary to reduce the size of input dataset so as to reduce the computational effort. As already mentioned, the technique used is the representation of the original image as byte average is based on a window size. The byte average varies from a value of 0 to 255. In [8] the optimum window size has been discovered in the range 256 to 1024 bytes. After few trials the default reduction window size implemented in this module was of 512 bytes as it gave the best results for the generated test data sets. This size can be varied by the user if required. The options which were explored to represent the byte average in the reduced disk image were:

- As integer value which is in the range 0-255; requires a maximum of three characters for representation.
- Normalization of integer byte value to the range 0-50; requires a maximum of two characters for representation.
- Hex values in precision 2; requires two characters for representation.

- ANSI character encoding; requires one character for representation.

Integer values including normalized ones did not result in optimum reduction sizes and the reverse mapping for extraction of data blocks was also a problem. The byte representation in hex of precision 2 helped in reverse mapping for data block extraction without affecting the reduction in size. The ANSI character set encoding leads to the most optimal reduction size as it outputs one character for each byte and also enables reverse mapping on to the original disk image from the graphical output so that data blocks of interest can be extracted. The reduced files are created in the same directory as those of input images and are not deleted as the same can be used as inputs for second iteration of similarity correlation. If one of the disk image inputs of the program is '200MBimg1.dd' then its reduced file is created as '200MBimg1.ddRN'. This reduced file is used for the NCD calculations.

### 5.3 Implementation of NCD Correlation Module

This module gives a choice of bzip2 or zlib compression for calculating the Normalized Compression Distance similarities between the disk images. The NCD is calculated block per block where the block size is the comparison window size. The window slides by one full size each time. A correlation scoring function  $S_{NCD}$  was devised to identify pairs of disk images with maximum similarities. The correlation score is calculated pair wise and reported based on certain fixed detection thresholds of NCD values. Based on these thresholds the scoring function  $S_{NCD}$  of disk image 1 and disk image 2 would be

$$S_{NCD} (Img1,Img2) = \frac{\sum_{j=N1}^{j=N1} (1-X_{1j}) + \sum_{j=N2}^{j=N2} (1-X_{2j}) + \dots + \sum_{j=Nn}^{j=Nn} (1-X_{nj})}{\text{Min} \{ \text{Img1.Blocks}, \text{Img2.Blocks} \}} \dots\dots\dots(5.3.1)$$

$S_{NCD}$  is a positive value if  $N1=N2\dots=Nn \neq 0$  else  $S_{NCD} (Img1,Img2)=0$ . Here  $Img1.blocks = (Img1.size) / (Comparison\ window\ size)$ .  $N1, N2$  etc are the total instances of NCD values within the corresponding thresholds.  $X1, X2$  etc are the NCD values within the corresponding thresholds. The individual values are subtracted from 1 to negate the effects of summation terms of large threshold values (low similarity) over small threshold values (high similarity) which may lead to incorrect inferences. The term in denominator  $Min \{Img1.Blocks, Img2.Blocks\}$  normalizes and bounds the  $S_{NCD}$  values.

The reason why this value in the denominator is sufficient is clear from Table 5.1. In case of similarity correlation of file x with itself, the maximum correlation ideally would be for total number of blocks in file x depending on the comparison window size. In some cases the similarities may be little more than the total number of blocks for a number of reasons. For example, in a text file the 'Introduction' at the start may be quite similar to the 'Conclusion' at the end. We may also come across a situation where one of disk image has multiple copies of files or data and hence the number of blocks similar would be more if these are found in the other drive image. In essence we are computing the percentage of blocks of the smaller drive image which are similar to the blocks in the other drive image.

Block m	0.8	0.9	0.7	0.82	0.99	1	1	0.2
	0.9	0.88	1	1	1	1	0.2	1
	0.25	0.89	0.99	0.89	0.88	0.2	1	1
File X	1	0.3	0.99	1	0.2	0.88	1	0.99
	0.8	0.9	0.99	0.2	1	0.89	1	0.82
	0.8	0.92	0.2	0.99	0.99	0.99	1	0.7
	0.9	0.2	0.92	0.9	0.3	0.89	0.88	0.9
Block 0	0.2	0.9	0.8	0.8	1	0.25	0.9	0.8
	Block 0	File X						Block m

Table 5.1: NCD values of File X with itself

The value  $NCD ( BlockX, BlockX )$  actually depends on the comparison window or block size. Greater the value of  $S_{NCD}$  greater is the correlation. Normally  $S_{NCD}$  would takes values from 0 to 1. We will get a value of 1 when the smaller object of comparison is fully contained in the other object and the NCD values are zero meaning that the blocks are perfectly similar. In this case we need to consider that there are no repetitions or duplicate file objects.

To illustrate the detection thresholds, we can use the following three detection thresholds of NCD values for a comparison window size of  $2k$  ; *Threshold 1:  $TH1 = [0, 0.3]$  , Threshold 2:  $TH2 = (0.3, 0.35]$  and Threshold 3:  $TH3 = (0.35, 0.4]$* . If there are  $m$  images then the results can be computed as a  $m \times m$  distance matrix. Correct selection of thresholds would lead to better interpretable graphical results.

As the NCD calculation process is comparison iteration of each block of file 1 with each block of file 2, the following simple structure was used to avoid repeated calculations.

```
for each block of file 1 do
{
    calculate and store file1_block_compression_size ;

    for each block of file 2 do
    {
        if first iteration then
        {
            calculate and store all file2_block_compression_size;
            concatenate blocks and calculate compression_size;
            calculate NCD;
        }
        else
        {
            concatenate blocks and calculate compression_size;
            calculate NCD using stored values;
        }
    }
}
```

## 5.4 Outputs and Data Blocks Extraction

The report of the pairwise NCD similarity comparison in form of correlation score  $S_{NCD}$ , input disk images, time of start, time of completion, NCD comparison window size, Reduction window, Thresholds and the compression algorithm used are part of a 'stats.txt' output file. The time taken also includes the disk image preprocessing time. The total count of points designated as similar is also mentioned as 'Count'. A sample 'stats.txt' file is shown below. Additional details of NCD values and the corresponding block number were also printed initially but were later removed as the details were felt unnecessary.

```
H:\Documents and Settings\Administrator\Desktop\files\200MBimg2.ddRN
H:\Documents and Settings\Administrator\Desktop\files\240offset1.ddRN
6/8/2008 7:45:09 PM
NCD BLOCK_SIZE 100
BZip2
Reduction window 50
Thresholds 0.35 0.4 0.5 0.55 0.6
6/8/2008 7:45:16 PM Completed Sncd 6.68286035161975E+0000 Count:1197
```

The graphical output is a 2 dimensional plot with left and bottom axes representing two disk images in terms of total blocks based on the comparison window size. Proper thresholds are selected based on the comparison window to get a clear picture of the similarities. A sample graph output is shown in Figure 5.1

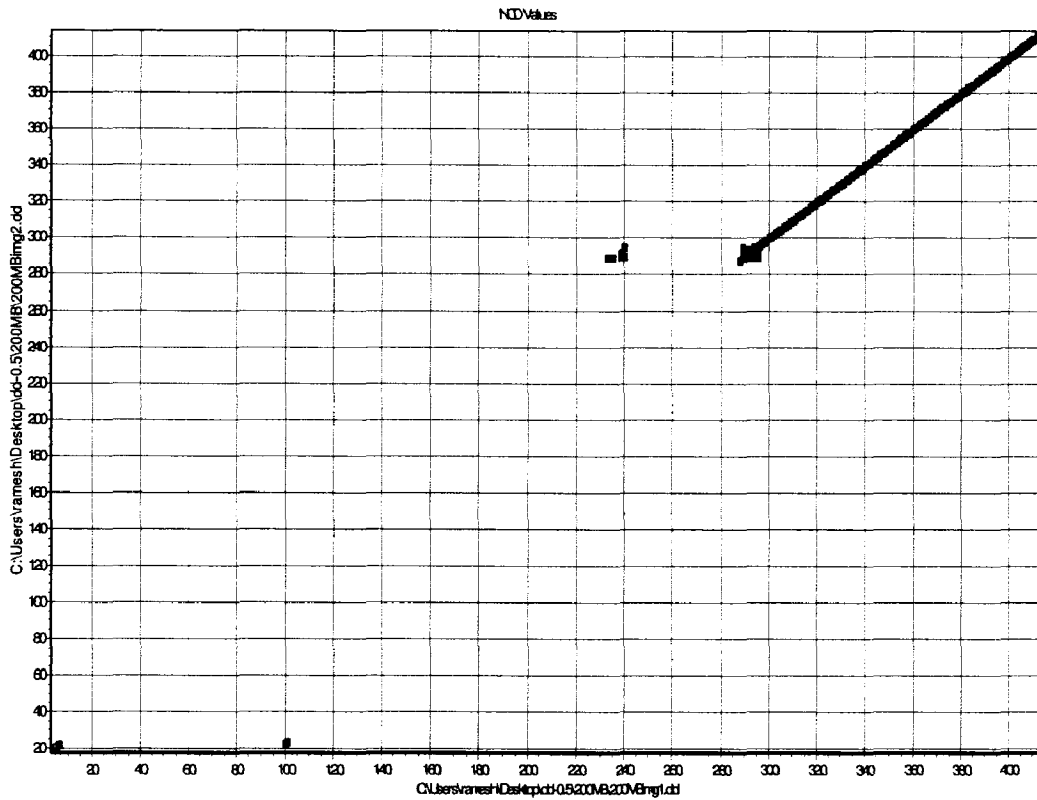


Fig 5.1: Correlation Graph: Plot of NCD values of two Disk images

For the purpose of extraction of data blocks matching similarity criteria, a simple mapping formula was used. For extracting the data of disk image of x-axis, the following equations were used to decide extraction positions:

$$\text{Start byte offset} = x_1 * R/C * W \dots\dots\dots(5.4.1)$$

$$\text{End byte offset} = x_2 * R/C * W \dots\dots\dots(5.4.2)$$

Where

- $x_1$  - X-coordinate of starting block
- $x_2$  - X-coordinate of ending block
- R - Reduction window size in bytes
- W - NCD comparison window size in bytes
- C - Number of encoding characters per byte used during reduction



## 5.5 Generation of Test Images

Raw test images were generated using the Unix 'dd' utility. For example the following sequence of commands using a Windows version of 'dd', a raw image *200MBimg3.dd* of 200MB with one known jpg file was generated. This file was also included as known similarity in one of the other images created.

```
C:\Users\ramesh\Desktop\dd-0.5>dd if=\\?\Device\HarddiskVolume5
bs=1M count=35 > 200MBimg3.dd
rawwrite dd for windows version 0.5.
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by the GPL. See copying.txt for details
35+0 records in
35+0 records out
```

```
C:\Users\ramesh\Desktop\dd-0.5>dd if=IMG_0313.JPG >>
200MBimg3.dd
rawwrite dd for windows version 0.5.
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by the GPL. See copying.txt for details
2698+1 records in
2698+1 records out
```

```
C:\Users\ramesh\Desktop\dd-0.5>dd if=\\?\Device\HarddiskVolume5
bs=1M count=175 skip=35 >> 200MBimg3.dd
rawwrite dd for windows version 0.5.
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by the GPL. See copying.txt for details
175+0 records in
175+0 records out
```

**6.1 Accuracy and Speed of Similarity Detection**

As we are dealing with large number of varying high capacity digital storage devices, it becomes imperative to produce accurate and quick results. Results which are not in-time will have very less value for the investigator and would also hamper the other aspects of the investigations and criminal proceedings. Similarly, accuracy of results is very important to avoid inconsistent conclusions and false implications. Below we discuss how both these requirements are satisfied in the NCD-cross drive correlation strategy.

**6.1.1 Window Size of Data Reduction**

In [8] the optimum window size has been discovered in the range 256 to 1024 bytes for file localization in steganalysis. This was so because a size greater than 1024 would miss out the original features and in some cases a size smaller than 256 may introduce unnecessary finer details. For NCD similarity correlation, reduction window greater than 1024 is bound to smoothen out certain features and therefore would miss out certain similarities of smaller size. It is also clear that a data object of size 1024 bytes may not be detected as the data object is just a character after reduction. Window sizes less than 256 would lead to more similarities but may not be useful always. For example header information of two pdf files would lead to a good similarity but the file contents can be very different. The accuracy of the NCD correlation would depend on the extent to which the characteristics of the raw data are captured in the reduced image.

Trials with 256k, 512k and 1024k reduction window sizes were conducted. However, most of the work was validated using 512k window size as this produced quick and reliable result for the test drive images generated as part of this work.

### **6.1.2 Window Size for Similarity Comparison**

Selection of the optimum window sizes for both reduction in preprocessing phase and computation of NCD are vital. The window sizes of 1k, 2k and 4k for NCD calculations between reduced disk images were tried out. It is obvious that short span similarities would be missed out as the window size increases. However, the effort increases exponentially as the size is halved. It was found that 2k window size for NCD calculations is optimum in the experiments. With more computational resources the NCD window can be decreased to 1k or below for better results. In one case of testing, a similarity of 400K was not detected when reduction window was 512k and the NCD comparison window was 4k.

### **6.1.3 Compression Algorithm**

Better the compression better is the accuracy. The real world lossless compression algorithms bzip2 and zlib(gzip implementation) were used for calculating NCD in the experiments. These algorithms satisfy the properties of Idempotency, Monotonicity, Symmetry and Distributivity to a large extent. As bzip2 is block based, it is symmetrical. The Complearn[4] NCD toolkit has bzip2 and gzip as built-in choices and this was an indication that bzip2 can be used. Bzip2, because of its better compression ratio, gives better results especially when the comparison window is large. For comparison block sizes ranging from 750KB to 1MB and above, the result of comparison of 'x' with 'x' were inconsistent as shown in figure 6.1. The exact comparison block size where the results become inconsistent were seen to be dependent on the type of data ( binary code, ASCII text etc). All these

observations were made initially when the input images were not reduced. After the preprocessing block was introduced in the algorithm, the size of NCD window was brought down to 2KB-4KB range from 700KB-1MB range, due to reduction and hence any real world compression would have worked, however, bzip2 was used in the final trials. Bzip2 has also been tested successfully for heterogeneous data of different file types in [9].

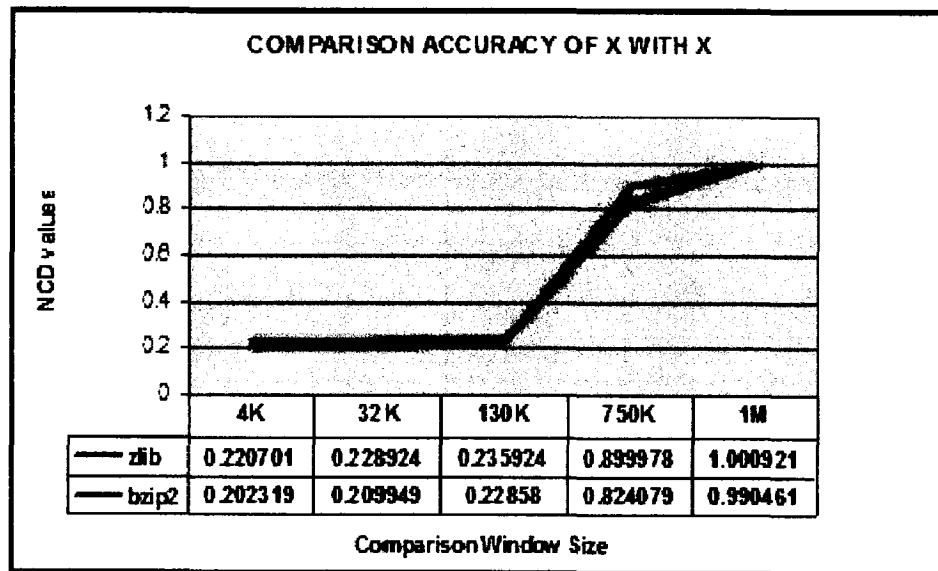


Fig 6.1: Comparison of Bzip2 and zlib Compression

#### 6.1.4 Random Noise Resistance of NCD

The Normalised Compression Distance is resistant to random noise to a large extent depending on the data type as shown in [11]. To be more precise, if some bits in one of the objects being compared are either replaced by random bits or the length is shifted few places, then the NCD value would not be affected much. What this means to our strategy is that even if two exactly similar data fragments on the disk images being compared, do not fit into same positions with respect to the NCD comparison window, as shown in figure 6.2, we will still detect a healthy degree of similarity. In figure 6.2 detection of 'aaaaaa' is not an issue. The data 'bbbbbb' of

first image file would be similar to third and fourth block of image 2 to varying degrees of similarity and hence detected despite the offset position.

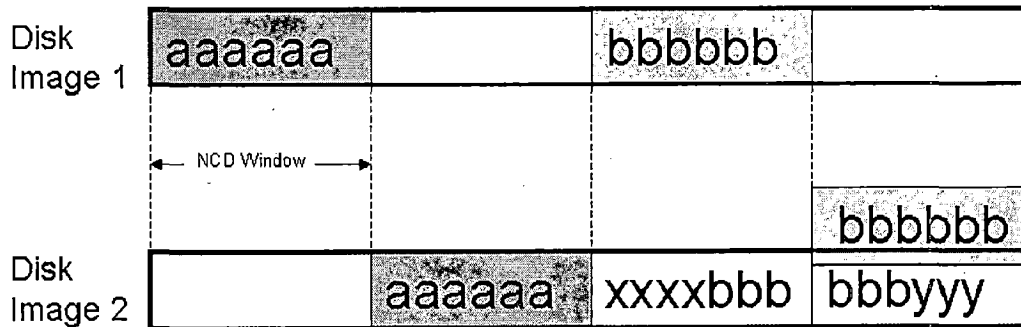


Fig: 6.2 : Sliding NCD Window's Resistance to Noise

Because of this property the comparison window need not move byte-per-byte. As long as the window size is small enough for the similarity sizes of interest, the iterations will lead to inevitable overlaps.

### 6.1.5 Effect of Relative Offsets of Data

The problem of detection misses due to induced relative data offset in one of the disk images, was also encountered. *Normally this situation would not occur in reality.* This is because of the way the reduction process works. During reduction in the preprocessing phase, the data in each reduction window size is converted to byte average which in turn is converted into an ANSI character. The situation is depicted in figure 6.3. Assume 'bbbbbb' is translated to 'B'. The block 'bbbbbb' which is aligned to window boundary in disk image 1 is converted to 'B'. Though 'bbbbbb' is present in disk image 2 but because it is not aligned to reduction window boundary, the encoding of the windows of image 2 would lose the required data signature. There is no 'B' in reduced disk image 2. The reduction process sees 'xxxxbbb' and 'bbbyyy' as the window aligned data chunks in disk image 2.

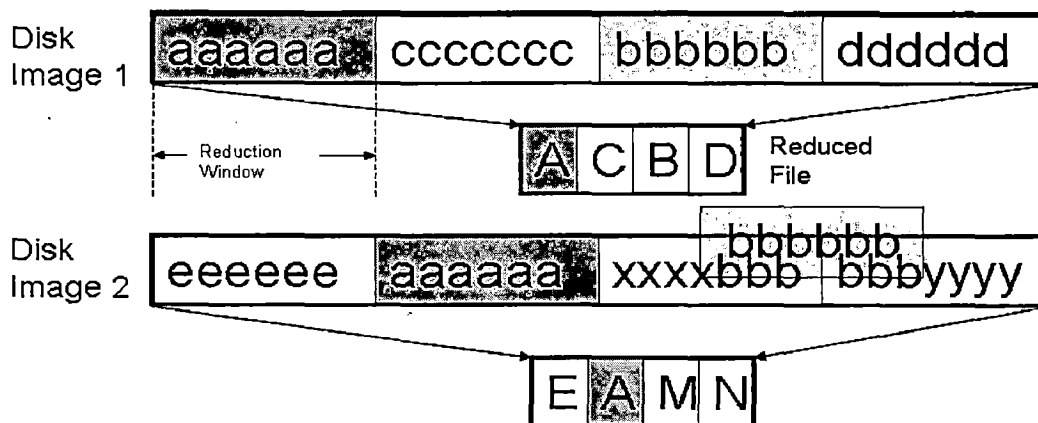


Fig 6.3: Problem of Offset in Reduction

If we consider a similar data fragment of one reduction window size then the maximum offset is half the size. In case the offset is exactly half we can decrease the reduction window size by half to preserve the data signatures. However, as the size of reduced file will double, the computational effort would increase many folds. It is also obvious similarities less than one reduction window would not have any unique signature. In the event of offset other than half window size, the reduction window size needs to be selected by trial and error so that its multiple is the amount of offset and hence neutralizes the effect. The other strategy might be addition of successively increasing filler bytes to one of the inputs and then selecting the iteration with maximum  $S_{NCD}$  or maximum 'diagonal lengths summation'. Such situations of offset may also exist due to certain file systems like 'ReiserFS' which does NOT follow sector based boundaries for allocating space to file objects and so offsets would be there at random positions. The mentioned strategies probably would not work here. *This issue of offset neutralization has not been dealt with in this dissertation though experiments were made to assess the effects of induced initial offsets.*

## 6.2 Optimization of Graphical Display

The NCD output is a jumble of cryptic numbers denoting varying degree of similarity between the data blocks. Understanding this output to find the correlation is very difficult. Hence a graphical output provides a simple and effective

visualization of the similarities between the two correlated images. Incorrect or out of band values may clutter the graph or miss out the similarities altogether. In case of optimized output consecutively correlated blocks appear as diagonals and give a clear picture to the investigator. The graphical display also forms the basis for extracting the correlated data blocks for further detailed analysis. Hence well optimized output will produce a better picture of comparison. The following two strategies of optimization were employed:

- NCD Thresholds
- Elimination of noise due to runs of zeros.

### 6.2.1 Threshold Values of NCD

Selection of Threshold values of NCD depends on the output Chart display requirements. The thresholds have no effect on the max  $S_{NCD}$  value selection as all  $S_{NCD}$  values would be equally affected. This optimization is different for different selection of comparison window size and has to be determined by trial and error. The thresholds used during testing, for 2k comparison window are:

```
if (ncd<=0.35)then show similarity as red
else if (ncd<=0.4) then show similarity as black
else if (ncd<=0.55) then show similarity as blue
else if ncd<=0.69 then show similarity as yellow
else if ncd<=0.75 then show similarity as lime ;
```

Figure 6.4(a) shows output with improper thresholds and figure 6.4(b) shows the optimized output.

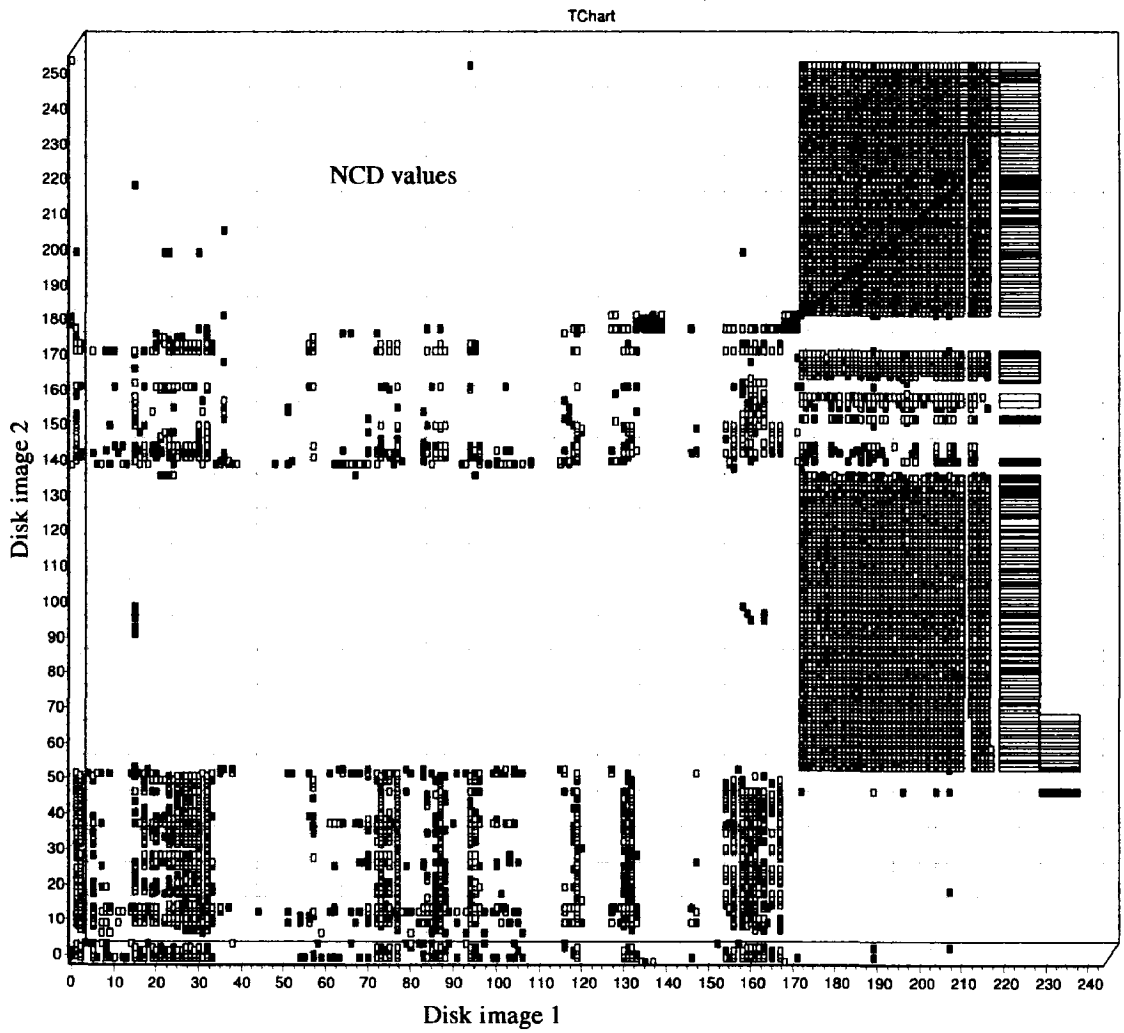


Fig6.4(a):NCD Values of two Disk Images with Incorrect Thresholds



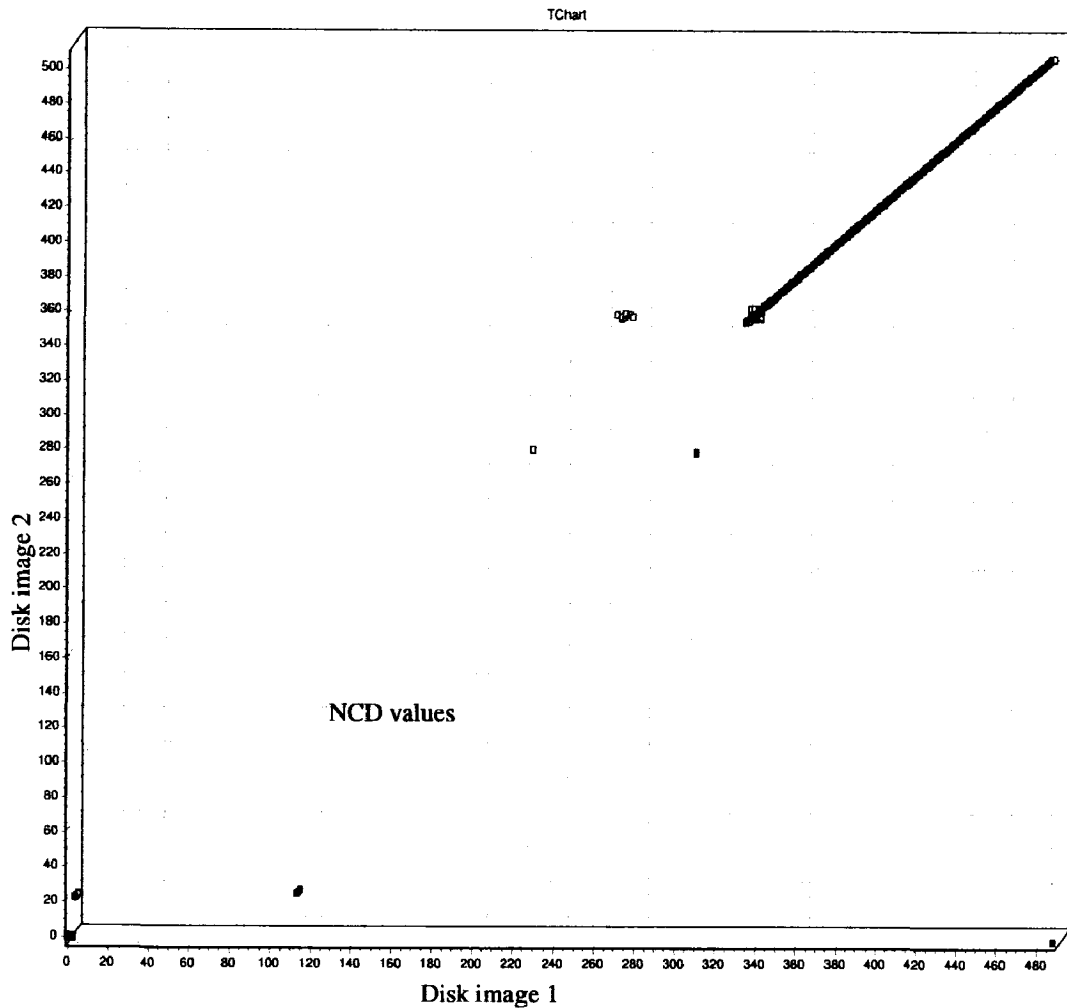


Fig 6.4(b): Values of two Disk Images with Optimized Thresholds

## 6.2.2 Noise Elimination during Data Reduction

Problems were encountered during NCD calculations when there were runs of zeros in the original images. Runs of zeros can also be attributed to some secure file wiping software and low level formatting of media. This resulted in false positives and clutter on the charts. *This noise is different from the random noise mentioned in paragraph 6.1.4.* This was overcome by allowing only runs of 4 consecutive zeros as byte averages and substituting the rest with random values, for the

reduction window size of 512 bytes. This optimization was also achieved by trial and error. However, it was noticed that the effect varies as the comparison window and reduction window sizes change. This noise can also be due to consecutive runs of some other byte average value in both the comparison objects but such likelihood is very small. Figure 6.5(a) shows the effect of noise and figure 6.5(b) shows the output after noise elimination.

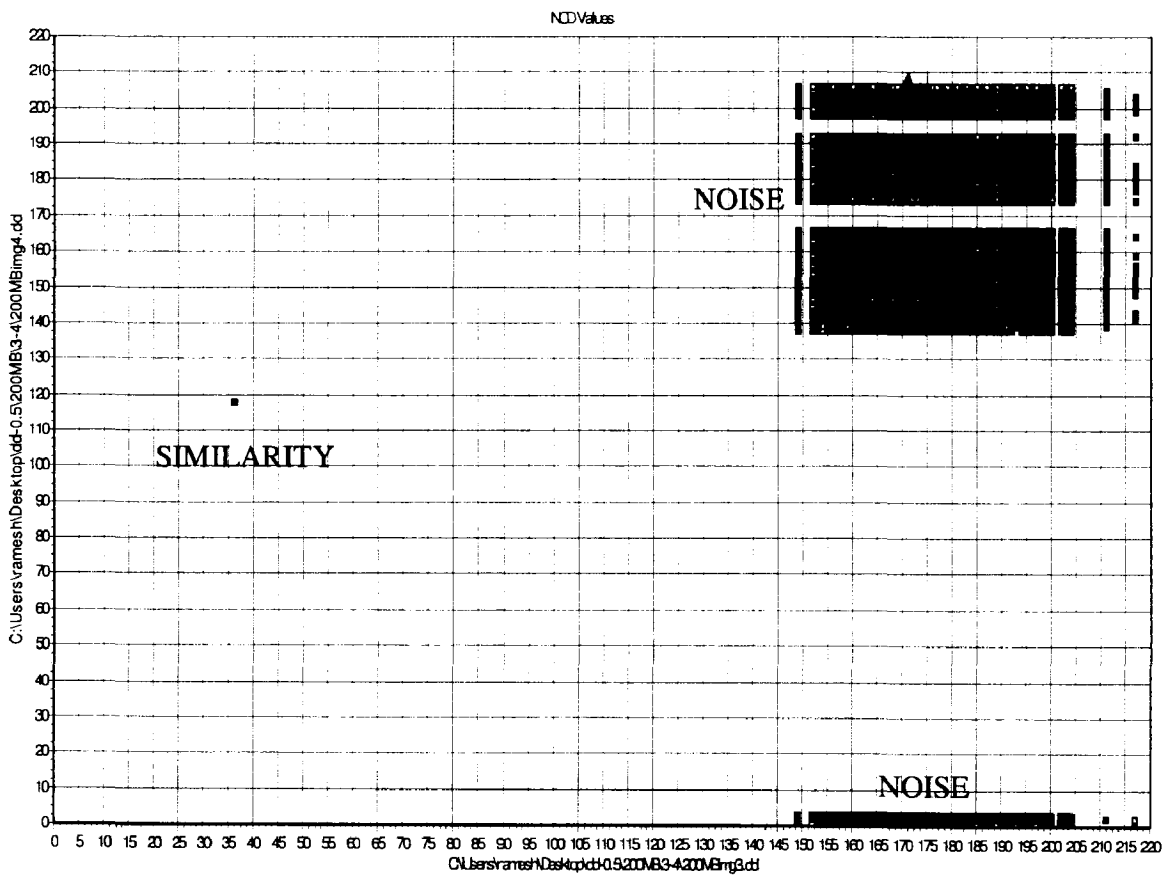


Fig 6.5(a): NCD Correlation with Noise

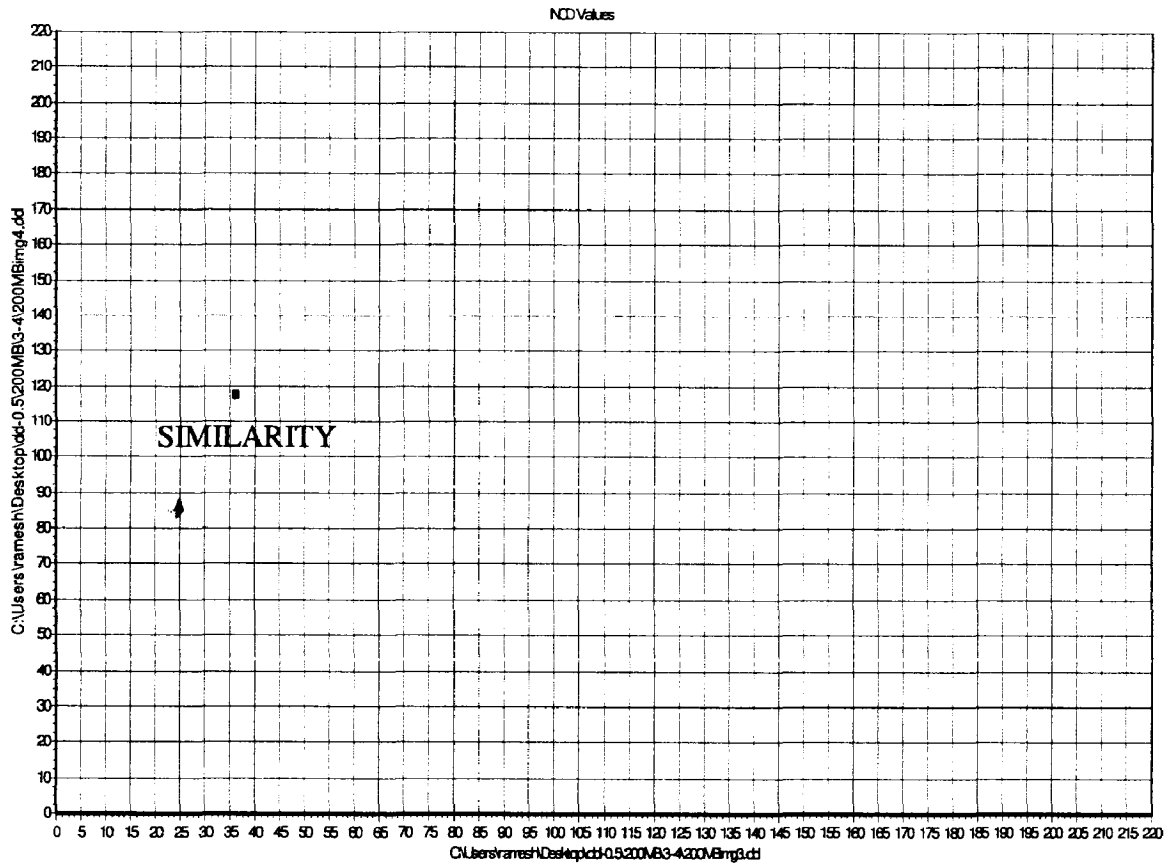


Fig 6.5(b): NCD Correlation after Noise Reduction

### 6.3 Limitations

The advantages have been enumerated in sufficient details. However, it is essential to understand the weaknesses. The limitations which became apparent during this dissertation are:

- Limitation of minimum size of similarity that can be detected. The comparison and the reduction windows play a role here as already discussed. However, decreasing the windows to detect small similarities would make increase the computational efforts many folds.
- Limitation due to different data storage formats. There are various formats for data storage. The most common of these are ASCII and

Unicode[7]. ASCII uses one byte to store characters whereas Unicode would use 4 bytes(UTF-32) or 2 bytes(UTF-16) or a variable number of 1, 2 or 4 bytes(UTF-8). Therefore the same file in two different formats, if not taken care of in the ‘Collection’ phase of forensic process, would not be detected in the raw format. The issue is further compounded because multiple byte representations can be of big-endian or small-endian order.

#### 6.4 Validation of Test Results

Disk images of size 200MB were created and used for validation of the algorithm. Deliberate similarities were introduced, deleted and then overwritten with other files so as to simulate deleted and slack space. A subset of the trials is produced below.

DISK IMAGE	IMG 1	IMG 2	IMG 3
IMG 2	0.1919251	-	-
IMG 3	0	0.0024886	-
IMG 4	0	0.0019140	0.0024253

Table 6.1:  $S_{NCD}$  Correlation Scores of 4 Test Images

Table 6.1 shows the results of NCD-cross drive correlation of four disk images. The comparison window used was 2KB size and the reduction window was of 512 bytes. Compression used was bzip2. The results were verified by subjecting the extracted data blocks to further analysis using WinHex(hex editor) and Scalpel(file carving) tools. The  $S_{NCD}$  values were as expected and the max value of 0.1919251 was due to large similarities introduced in Img1 and Img2. A jpg file of 900 KB was injected in Img3 and Img 4. The value of  $S_{NCD}(Img2,Img3)$  as 0.0024886 was not due to introduced similarities but was due to residual file fragments. The other values are indication of insignificant or no correlation. Different color schemes were used to indicate the various threshold ranges in the output chart. Similarities greater than 1 window were indicated as diagonals.

### **7.1 Conclusion**

Cross drive analysis by correlation of evidence spanning multiple digital devices with ever increasing capacities would be an important factor of digital investigations in future. Techniques such as these would address some of the many challenges that are faced in digital investigations [12, 13]. In this work it was shown that NCD can be used in such scenarios for parameter free correlation of the disk images, which inherently has many advantages as already elaborated. It would be possible to quickly highlight all hot drives or devices and the strongest relations amongst the drive images. This information would provide the necessary impetus to the investigation. The algorithm was validated in lab conditions and owing to time and other constraints the sizes of the drive images were restricted to 200MB. However, the program developed can be easily used for drive images of larger size. Correlation time of about 3 - 4 min was achieved with NCD window size of 2k for 200MB drive images reduced with reduction block size of 512 bytes. Considering these statistics, for correlating two 1GB drive images the time required on a normal present day desktop would be  $3 \times (5 \times 5) = 75$  minutes. This time is quite reasonable even without optimization and other enhancements. Individual analysis of the drives using data carving, key word search etc would take many hours. The calculation of NCD correlation scores of pairwise images would take insignificant fraction of time and hence can be disregarded.

### **7.2 Scope for Future Work**

Before using the technique in field, further extensive experiments with real data sets is essential. The system can be enhanced using cluster computing, grid computing,

multi-threading etc to deal with the large capacities and large numbers of digital devices. Another important area of work is the preprocessing of disk images. It is important to devise more effective reduction methods so that greater reduction is achieved while maintaining original data signature of the digital storage devices. The present reduction uses mapping of 1 byte to ANSI character set. By using 4 byte Unicode UTF-32 the reduction can be further enhanced by an order of four.

These two enhancements would work hand-in-hand for a fruitful application of this approach in digital forensics. Study of recursive NCD-cross drive correlation for obtaining faster results within the acceptable accuracy limits can also be undertaken.

## References

---

---

- [1] Eoghan Casey, "Digital Evidence and Computer Crime" second edition, Elsevier Academic Press, 2004.
- [2] Proise and Mandia, " Incident Response and Computer Forensics" ,second edition, McGraw Hill/Osborne, 2003.
- [3] Simson L. Garfinkel, "Forensic feature extraction and cross-drive analysis", Digital Investigation – Elsevier, DFRWS, 3S(2006)S71 – S81, 2006.
- [4] M. Li, X. Chen, X. Li, B. Ma, and P. Vitányi, "The similarity metric," IEEE Trans. Inf. Theory, vol. 50, no. 12, pp. 3250–3264, Dec. 2004.
- [5] Report From the First Digital Forensic Research Workshop (DFRWS), "A Road Map for Digital Forensic Research", DTR - T001-01 FINAL DFRWS TECHNICAL REPORT, November 6th, 2001.
- [6] NIST-Special Publication 800-86, "Guide to Integrating Forensic Techniques into Incident Response", National Institute of Standards and Technology, U.S. Department of Commerce, 2006.
- [7] Brian Carrier, "File System Forensic Analysis", Addison-wesley, 2005.
- [8] Robert F.Erbacher, John Mulholland, "Identification and localization of Data Types within Large-Scale File Systems", Proceedings of the Second International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE'07), IEEE, ISBN:0-7695-2808-2, pp. 55-70, 2007.



- [9] Rudi Cilibrasi and Paul M.B. Vitányi, “Clustering by Compression”, IEEE transactions on Information Theory 51(4), pp 1523-1545, 2005.
- [10] Eamonn, Stelo and Ratana, “Towards Parameter-Free Data Mining”, KDD proceedings 04, Aug 22-25, ACM, 2004.
- [11] Cebrian, Alfonseca and Ortega, “The Normalized Compression Distance is Resistant to Noise”, Digital Object Identifier 10.1109/TIT.2007.894669, IEEE, Revised Manuscript, 2007.
- [12] Golden G. Richard III and Vassil Roussev, “Next-Generation DIGITAL FORENSICS”, Communications of the ACM, Vol. 49: No. 2, pp 76-80, February 2006.
- [13] Panda and Giordano, “Next-Generation CYBER FORENSICS”, Communications of the ACM, Vol. 49: No. 2, pp 44-47, February 2006.

1. A paper “Application of Normalized Compression Distance for Cross Drive Analysis by Correlation” has been submitted to the Journal *DIGITAL INVESTIGATION*, *ELSEVIER* on 29 May 08. *Manuscript Number: DIIN-D-08-00013*.

# APPENDICES

```
{Delphi 7 code}

unit Unit1;           //This is the GUI form unit

interface

uses                 //Include files

    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
    Forms, Dialogs, StdCtrls, zlib, Grids, ExtCtrls, Series, Chart,
    Menus, math;

{/*-----
The NCD comparison window and the preprocessing reduction window size are
taken as inputs from the user. The max values which are also default values are
declared as constants. These values can be changed to suit our requirements.
-----*/}

Const
    MAX_SECTOR_SIZE=2000;           //Max NCD window
    MAX_RBLOCK_SIZE=512;           //Max reduction window

{/*-----
The following are the GUI components on the GUI Form
-----*/}

type
    TForm1 = class(TForm)
        OpenDialog1: TOpenDialog;
        ListBox1: TListBox;
```

Correlate: TButton;  
Memo1: TMemo;  
RadioGroup1: TRadioGroup;  
LZW: TRadioButton;  
Bzip2: TRadioButton;  
MainMenu1: TMainMenu;  
Add2: TMenuItem;  
clear2: TMenuItem;  
StartX: TEdit;  
StartY: TEdit;  
EndX: TEdit;  
EndY: TEdit;  
NCDWindow: TEdit;  
ReductionWindow: TEdit;  
GroupBox1: TGroupBox;  
GroupBox2: TGroupBox;  
Label1: TLabel;  
Label2: TLabel;  
Extract: TButton;  
Panel1: TPanel;  
Chart1: TChart;  
Series1: TPointSeries;  
Label3: TLabel;  
Label4: TLabel;  
Label5: TLabel;  
Label6: TLabel;  
Label7: TLabel;  
Label8: TLabel;  
Edit1: TEdit;  
Edit2: TEdit;  
Edit3: TEdit;

```
Edit4: TEdit;
Edit5: TEdit;
All1: TMenuItem;
Chart2: TMenuItem;
Defaults1: TMenuItem;
GroupBox3: TGroupBox;
Label9: TLabel;
ZeroLimit: TEdit;
```

```
{/*-----
The following are the form component event procedure declarations.
-----*/}
```

```
procedure CorrelateClick(Sender: TObject);
procedure Add2Click(Sender: TObject);
procedure ExtractClick(Sender: TObject);
procedure Chart2Click(Sender: TObject);
procedure All1Click(Sender: TObject);
procedure Defaults1Click(Sender: TObject);
```

```
private
    { Private declarations }
```

```
{/*-----
The following are the public(global) variable declarations
-----*/}
```

```
Public
    { Public declarations }
```

```
NCDCCompWin: Integer;           //NCD Comparison window size input
```

```

ReductionWin: Integer;           //Reduction widow size input

//Variable declarations for bytes read during file reads
Bytes_read: Integer;
Bytes_read2: Integer;

// file read loop variables
sc2: Integer;
sc2Copy: Integer;

ImgFileHandle: file of byte;     //File handle for first input object
ImgFileTwoHandle: file of byte;  //File handle for first input object

size1: int64;                    //Compression size of block of first file
size2: int64;                    //Compression size of block of first file
size3: int64;                    { //Compression size of block of first file
                                concatenated with block of second file}

ThSecCount: int64;              //NCD Bocks within thresholds
CompleteCount: int64;          //Total NCD blocks

//Variables for threshold instances
N1: int64;
N2: int64;
N3: int64;
N4: int64;
N5: int64;
N6: int64;
NN: int64;

```

```

size2Store : array[0..100000] of integer;    //array to store second file
                                              compression values}

firstiteration: boolean;
innerLoopRunComplete : boolean;
TempFileHdl: textfile;                      //report file handle
FirstFile: TMemoryStream;
SecFile:TMemoryStream;

NCD: extended;
ThresholdSc: extended;
WThresholdSc : extended;

```

**//Temp variables to store summation of (1-NCDvalue) within each Threshold**

```

X1 : extended;
X2: extended;
X3: extended;
X4: extended;
X5: extended;
X6: extended;
X7: extended;
X8 : extended;

```



**{//Buffers to store bytes read and the concatenated blocks}**

```

BufX : Array[0.. MAX_SECTOR_SIZE]of Byte;
BufX2 : Array[0.. MAX_SECTOR_SIZE] of Byte;
BufXCarve : Array[0.. MAX_SECTOR_SIZE *MAX_RBLOCK_SIZE] of
byte;

```

**//MemoryStreams used for compression**

```

ms1: TMemoryStream;
ms11: TMemoryStream;

```



```
ms2: TMemoryStream;  
ms22: TMemoryStream;  
ms3: TMemoryStream;  
ms33: TMemoryStream;  
msE: TMemoryStream;  
mstemp: TMemoryStream;
```

**//Procedure declarations**

```
procedure CompressStream(inpStream, outStream: TStream);  
procedure Reduce;
```

**//Function Declarations**

```
Function NCDvalue : extended;
```

```
end;
```

```
var
```

```
Form1: TForm1;
```

```
implementation
```

```
uses bzip2;
```

```
{ $R *.dfm }
```

```
{/*-----
```

**The following is the main procedure which performs correlation on the two input disk images. The sequence of tasks performed is:**

- 1. call preprocessing procedure 'Reduce' to reduce input files**
- 2. Loop through the reduced image files based on comparison window size and calculate NCD using function 'NCDvalue'**
- 3. Assign values to graph based on threshold**
- 4. calculate correlation score  $S_{NCD}$**

## 5. Output details in report 'stats.txt'

```
-----*/}

procedure TForm1.CorrelateClick(Sender: TObject);

var
    sc: integer;
    minCount: integer;
    MinBlocks: integer;
    Buf2total: integer;
    i: integer;
    j: integer;
    k : integer;
    TempFile: file of byte;

begin
    Memo1.Lines.Add(datetimetostr(now));           //Record date-time of start
    ReductionWin:=strtoint(ReductionWindow.text); //Read reduction window
                                                    input}
    NCDCompWin:=strtoint(NCDWindow.text);        //Read NCD window input
    //Set chart axes as "disk image file1" and "disk image file 2"
    chart1.BottomAxis.Title.Caption := ListBox1.Items.Strings[0];
    chart1.LeftAxis.Title.Caption := ListBox1.Items.Strings[1];

    {/-----
    Call to preprocessing procedure
    .....*/}
    Reduce;
```

```

{ /*-----
Assign input disk image file names to graph axes
-----*/}

chart1.BottomAxis.Title.Caption := ListBox1.Items.Strings[0];
chart1.LeftAxis.Title.Caption := ListBox1.Items.Strings[1];

{ /*-----
Initializations
-----*/}

sc:=0;
Buf2total:=0;
ThSecCount:=0;
CompleteCount:=0;
Bytes_read:=1;
X1:=0;
X2:=0;
x3:=0;
x4:=0;
x5:=0;
x6:=0;
x7:=0;
x8:=0;
N1:=0;
N2:=0;
N3:=0;
N4:=0;
N5:=0;
N6:=0;
NN:=0;
ThresholdSc:=0;

```

**{ The firstIteration Boolean variable is used to store the values of all Size2 so as to avoid repeat computations. This saves time}**

```
firstIteration:=true;
```

```
{/*-----  
Opens reduced files for reading  
-----*/}
```

```
AssignFile(ImgFileHandle, ListBox1.Items.Strings[0]+'RN');
```

```
AssignFile(ImgFileTwoHandle, ListBox1.Items.Strings[1]+'RN');
```

```
//Filemode:=0;
```

```
//showmessage(ListBox1.Items.Strings[0]);
```

```
{/*-----  
position file pointer at the start  
-----*/}
```

```
Reset(ImgFileHandle);
```

```
Reset(ImgFileTwoHandle);
```

```
{/*-----  
Creates a file 'stats.txt' and opens it for writing in append mode.  
The input disk image file names are written to this file.  
-----*/}
```

```
assignfile(TempFileHdl, 'stats.txt');
```

```
Rewrite(TempFileHdl);
```

```
//reset(TempFileHdl);
```

```
append(TempFileHdl);
```

```
writeln(TempFileHdl, ListBox1.Items.Strings[0]);
```

```
writeln(TempFileHdl, ListBox1.Items.Strings[1]);
```

```

{ /*-----
Either zlib or bzip2 compression algorithm is selected based on users choice.
Default selection is bzip2.
These are noted in the 'stats.txt' file along with date and time.
The threshold values are also written to 'stats.txt'
-----*/}

```

```

if    LZW.Checked                //Chk selection of compression algo
    then
        writeln(TempFileHdl,DatetimeToStr(now),' NCD
        BLOCK_SIZE ',NCDCCompWin,' LZW',' reduction window
        ',ReductionWin)
    else
        writeln(TempFileHdl,DatetimeToStr(now),' NCD
        BLOCK_SIZE ',NCDCCompWin,' BZip2',' reduction window
        ',ReductionWin);

```

```

writeln(TempFileHdl,' Thresholds ',edit1.text,' ',edit2.text,' ',edit3.text,' ',edit4.text,'
',edit5.text);                //Thresholds
writeln(TempFileHdl," );

```

```

{ /*-----
Bytes equal comparison window size are read into buffer from first reduced disk
image file in the outer while loop. The inner while loop reads bytes as per
window size into buffer from the second reduced file.
-----*/}

```

```

while ( Bytes_Read >0) do                // Outer while loop
    begin
        Bytes_read2:=1; sc2:=0;//I:=0
        firstiteration:=true;

```

```

{Set file position}
Seek(ImgFileHandle, (sc * NCDCompWin));
{read bytes of size comparison window into BufX}
BlockRead(ImgFileHandle, BufX, NCDCompWin, Bytes_Read);

while (Bytes_Read2 >0) do                                     //Inner while loop
begin
    {Set file position}
    seek(ImgFileTwoHandle,(sc2* NCDCompWin));
    {read bytes of size comparison window into BufX2}
    BlockRead(ImgFileTwoHandle, BufX2, NCDCompWin,
Bytes_Read2);
    //SecFile.Seek(sc2 * SECTOR_SIZE, soFromBeginning);
    //secFile.ReadBuffer(bufx2,bytes_read2);

    {call function to calculate Normalized Compression Distance and
store the value in var ncd and perform the necessary calculations}
    ncd=NCDvalue;
    writeln(TempFileHdl,' NCD ',floattostr(ncd),' #',sc,' &',sc2,' size 1 2
3:',size1,' : ',size2,' : ',size3);

    if ncd<2 then
begin
        CompleteScore:=CompleteScore+ncd;
        inc(CompleteCount);
end;

    if(ncd<StrToFloat(Threshold.Text)) then
begin
        writeln(TempFileHdl,' NCD ',floattostr(ncd),' #',sc,' &',sc2);

```



```

end
else if (ncd<=( StrToFloat(Edit3.Text))) then
begin
    with form1.Series1 do AddXY( sc, sc2, ",clblue );
                                //Add point to graph
    x3:=x3+(1-ncd); //Sum all NCD values in this Th
    inc(N3); //Update number of instances
end
else if ncd<= (StrToFloat(Edit4.Text))then
begin
    with form1.Series1 do AddXY( sc, sc2, ", clyellow );
                                //Add point to graph
    x4:=x4+(1-ncd); //Sum all NCD values in this Th
    inc(N4); //Update number of instances
end
else if ncd<= (StrToFloat(Edit5.Text))then
begin
    with form1.Series1 do AddXY( sc, sc2, ", cllime );
                                //Add point to graph
    x5:=x5+(1-ncd); //Sum all NCD values in this Th
    inc(N5); //Update number of instances
end;

```

```

{ /*-----
For a comparison window size of 1000k, five threshold levels are used to
generate graph and for the correlation score  $S_{NCD}$ . The variables used in the
score calculation are updated.
-----*/}

```

```

if SECTOR_SIZE=1000 then

```



```

if (ncd<=0.2)then
begin
    with form1.Series1 do AddXY( sc, sc2, ", clred );
                                //Add point to graph
    x1:=x1+(1-ncd); //Sum all NCD values in this Th
    inc(N1);           //Update number of instances
end
else if (ncd<=0.25) then
begin
    with form1.Series1 do AddXY( sc, sc2, ",clblack );
                                //Add point to graph
    x2:=x2+(1-ncd); //Sum all NCD values in this Th
    inc(N2);           //Update number of instances
end
else if (ncd<=0.3) then
begin
    with form1.Series1 do AddXY( sc, sc2, ",clblue );
                                //Add point to graph
    x3:=x3+(1-ncd); //Sum all NCD values in this Th
    inc(N3);           //Update number of instances
end
else if ncd<=0.4 then
begin
    with form1.Series1 do AddXY( sc, sc2, ", clyellow );
                                //Add point to graph
    x4:=x4+(1-ncd); //Sum all NCD values in this Th
    inc(N4);           //Update number of instances
end
else if ncd<=0.65 then
begin
    with form1.Series1 do AddXY( sc, sc2, ", cllime );

```

```

//Add point to graph
x5:=x5+(1-ncd);//Sum all NCD values in this Th
inc(N5); //Update number of instances
end;

//ThresholdSc:=ThresholdSc+ncd; inc(ThSecCount);
end;

inc(sc2);
firstiteration:=false;
end;

inc(sc);
innerLoopRunComplete:=true;
end;

if sc>sc2 then MinBlocks:=sc2
else MinBlocks:=sc;
{ /*-----
NN is the total of instances within the various thresholds
-----*/}
NN:=(N1+N2+N3+N4+N5) ;

{ /*-----
The reduced files file handles are freed and files closed
-----*/}
closefile(ImgFileHandle);
closefile(ImgFileTwoHandle);

```

```

Memo1.Lines.Add('COMPLETE'+inttostr(n1)+' '+inttostr(n2)+' '+inttostr(n3)+'
'+inttostr(n4)+' '+inttostr(n5)+'xx'+inttostr(nn));
Memo1.Lines.Add('DONE-'+datetimetostr(now));
{ /*-----
The correlation score  $S_{NCD}$  is calculated
-----*/}

//if ThSecCount>0 then ThresholdSc:=ThresholdSc/ThSecCount;
if NN>0 then WThresholdSc:=(x1 + x2 + x3 + x4 + x5)/(MinBlocks);

{ /*-----
Write the score and other details to 'stats.txt'
-----*/}

writeln(TempFileHdl,DatetimeToStr(now),' Completed ', Sncd ',WThresholdSc,'
Count:',NN);
Closefile(TempFileHdl);

{ /*-----
Save output graph as a file
-----*/}

Chart1.SaveToMetafile('chart.wmf');

end;                                // End of procedure correlate.click

/*-----
This procedure 'Reduce' performs the preprocessing on the input disk image
files.
All the file items in the Listbox are reduced one by one.
-----*/}

procedure TForm1.Reduce;
var

```

```
TotalFiles: integer;
ii: integer;
i,j:integer;
Val_total: integer;
Sc:integer;
Bytes_Read: integer;
ZeroCount:integer;
Reducedfile :textfile;
name1,name2:string;
average:double;
twoByte: array[0..1] of byte;
DoNow : boolean;
```

```
begin
```

```
TotalFiles:=ListBox1.Items.Count;
```

```
ii:=0;
```

```
randomize;
```

```
while ii< (TotalFiles) do
```

```
begin
```

```
DoNow:=true;
```

```
{Open input file for reading}
```

```
AssignFile(ImgFileHandle, ListBox1.Items.Strings[ii]);
```

```
Reset(ImgFileHandle);
```

```
{Create and open file for writing reduced data}
```

```
AssignFile(Compressfile,ListBox1.Items.Strings[ii]+'RN') ;
```

```
rewrite(Reducedfile);
```

```
{Initializations}
```

```
sc:=0;
```

```

Bytes_Read:=1;
ZeroCount:=0;
while ( Bytes_Read >0) do
begin
    Seek(ImgFileHandle, (sc * ReductionWin));
    BlockRead(ImgFileHandle, BufX, ReductionWin,
    Bytes_Read);
    Val_total:=0;
    if Bytes_Read>0 then
    begin

        {Calculate Byte average for the reduction window}
        for i := 0 to Bytes_Read do
        begin
            Val_total:=Val_total+BufX[i];
            //Average:= Val_total div (i+1);
        end;
        //DoNow:=Not(doNow);
        //Wbuf[0]:= Val_total div (i+1);
        Average:= Val_total / (i+1);
        //Average:=(((Average-MIN)/(MAX-
        MIN))*(NEW_MAX-NEW_MIN))+NEW_MIN;
        //showmessage( inttostr(Bytes_Read)+'-'+
        inttohex(round(Average),2));
        //write( Compressfile1,inttohex(round(Average),2));
        if round(Average)=0 then inc(ZeroCount)
            else ZeroCount:=0;

        {Encode byte rounded average as ANSI character
only if there are no 4 consecutive runs of zero byte

```

```

        average else write a random ANSI character}

        if ZeroCount<4 then write (Reducedfile,
        char(round(Average)))
            else write(Reducedfile,
            char(randomrange(0,255)));

        end;

        inc(sc);
        end;

    Bytes_Read:=1;sc:=0;

    {close file handles}
    closefile(ImgFileHandle);
    closefile(Reducedfile);

    inc(ii); // Increment loop variable
    end;

Memo1.Lines.Add('Reduced');
end; // End of procedure 'Reduce'

{/*-----
    Compression procedure using either zlib or bzip2 units.
    Data is compressed using memory streams.
    -----*/}

```

```

procedure TForm1.CompressStream(inpStream, outStream: TStream);
var
    ZStream: TCustomZLibStream;
    BZStream: TCustomBZip2Stream;
begin
    try
        if LZW.Checked then
            begin
                //showmessage('LZW');
                {For zlib we use the max compression to get better results}
                ZStream := TCompressionStream.Create(c1Max, OutStream);
                ZStream.CopyFrom(inpStream, 0);

            end;
            if Bzip2.Checked then
                begin
                    {For bzip2 compression a block size of 900k is used}
                    BZStream:=TBZCompressionStream.Create(bs9,OutStream);
                    BZStream.CopyFrom(inpStream,0);
                end;

            finally

                if LZW.Checked then ZStream.Free;
                if Bzip2.Checked then BZStream.Free;

            end;

        end;
    end;
end;

```

```

{ /*-----
Function to calculate Normalised Compression Distance
-----*/}

Function TForm1.NCDvalue : extended;
begin
if firstiteration then
try
    ms1 := TMemoryStream.Create;
    ms1.Clear;
    ms1.WriteBuffer(bufx,Bytes_read);
    ms11 := TMemoryStream.Create;
    CompressStream(ms1, ms11);
    //size1:=0;

{ /*-----
The compressed block size of block of first file is saved in variable 'Size1' so that
need not be computed again, until all the blocks of second file have been
compared. Boolean variable 'firstiteration' used to check the status.
-----*/ }

    size1:=ms11.Size;
    ms3 := TMemoryStream.Create;
    ms3.WriteBuffer(bufx,Bytes_read);
    // writeln(TempFileHdl,'-----calculated ');
finally
    ms1.Free;ms11.Free;
    firstiteration:=false;
end
else
begin
    ms3 := TMemoryStream.Create;

```



```

    ms3.WriteBuffer(bufx,Bytes_read);
end;

{ /*-----
Compressed block sizes of blocks of the second file are calculated only once and
stored in array size2Store by using the Boolean variable
'innerLoopRunComplete'. This method saves time and effort.
-----*/}

if innerLoopRunComplete=false
then
try
    ms2 := TMemoryStream.Create;
    ms2.Clear;
    ms2.WriteBuffer(bufx2,Bytes_read2);
    ms22 := TMemoryStream.Create;
    ms22.Clear;
    CompressStream(ms2, ms22);
    //size2:=0;
    size2:=ms22.Size;
    size2Store[sc2]:=ms22.Size;
    //writeln(TempFileHdl,'inner loop compress');
finally
    ms2.Free;ms22.Free;
    ms3.WriteBuffer(bufx2,Bytes_read2);
end
else
begin
    size2:=size2Store[sc2];
    //writeln(TempFileHdl,size2);
    ms3.WriteBuffer(bufx2, Bytes_read2);
end;

```

```
/*-----  
The blocks of file1 and file 2 are concatenated and their compressed lengths are  
calculated using streams m3 and m33  
-----*/
```

```
try  
    ms33 := TMemoryStream.Create;  
    ms33.Clear;  
    CompressStream(ms3, ms33);  
    size3:=ms33.Size;  
    //writeln(TempFileHdl,size3);
```

Finally

```
    ms3.Free;  
    ms33.Free;  
end;
```

**{calculation of Normalised Compression Distance}**

```
if    (size1>size2)  
    then  NCDvalue:=(size3-size2)/size1  
    else  NCDvalue:=(size3-size1)/size2 ;  
  
end;
```

```
/*-----  
Input file 'add' click event for taking disk image inputs  
-----*/
```

```
procedure TForm1.Add2Click(Sender: TObject);
```

```

begin
  OpenFileDialog1.Execute;
  ListBox1.Items.Add(OpenFileDialog1.FileName);
  innerLoopRunComplete:=false;
  sc2copy:=0;
end;

```

```

{ /*-----
Procedure to clear input selections and display graph
-----*/ }

```

```

procedure TForm1.All1Click(Sender: TObject);
begin
  ListBox1.Clear;
  Series1.Clear;
end;

```

```

{ /*-----
Procedure to cleardisplay graph
^
-----*/ }

```

```

procedure TForm1.Chart2Click(Sender: TObject);
begin
  Series1.Clear;
end;

```

```

{ /*-----
Procedure to default values
-----*/ }

```

```

procedure TForm1.Defaults1Click(Sender: TObject);

```

begin

NCDWindow.Text:='2000'; //2000k NCD window  
ReductionWindow.Text:='512'; //512k Reduction window

**//Threshold for NCD window 2000k**

Edit1.Text:= '0.35';

Edit2.Text:= '0.4';

Edit3.Text:= '0.5';

Edit4.Text:= '0.69';

Edit5.Text:= '0.8';

**//For zero run noise elimination; max zero average values default setting**

ZeroLimit.Text:='4';

end;

*/\*-----\**

**Procedure to extract blocks matching similarity criteria.**

**The co-ordinates of points on graph are used to calculate the file positions**

*-----\*/*

procedure TForm1.ExtractClick(Sender: TObject);

var

ExtractFileHandle: file of byte;

ExtractFileTwoHandle: file of byte;

```

CarveOffset : Longint;
OutputFileHandle : Integer;
i: Integer;
Bytes_Written: Integer;
BytesRead,count : Integer;

begin

    {open input raw files}
    AssignFile(ExtractFileHandle, ListBox1.Items.Strings[0]);
    AssignFile(ExtractFileTwoHandle, ListBox1.Items.Strings[1]);

    {Position pointers at the start of file}
    Reset(ExtractFileHandle);
    Reset(ExtractFileTwoHandle);

    {Create and open files to store data blocks of first file}
    OutputFileHandle := FileCreate(StartX.Text + '-' + StartY.Text + 'FIRST' +
    EndX.Text + '-' + EndY.Text);

    {Position the file pointer for carving the data}
    FileSeek(OutputFileHandle,0,2);
    count:= NCDCompWin*ReductionWin;

    {carve data from the start point to end point on the x-axis of graph}
    for i:=strtoint(StartX.Text) to strtoint(EndX.Text) do
    begin

        // Set file position
        //CarveOffset := ((i) * (SECTOR_SIZE div 2)*512 );
        CarveOffset := ((i) * count;

```

```

Seek(ExtractFileHandle, CarveOffset );

// Read data
BlockRead(ExtractFileHandle, BufXcarve, count, BytesRead);

//write carved data to output file
FileWrite(OutputFileHandle, BufXcarve, BytesRead);

end;

//Close output file handle
FileClose(OutputFileHandle);

{ Create and open files to store data blocks of second file}

OutputFileHandle := FileCreate(StartX.Text + '-' + StartY.Text + 'SECOND'
+ EndX.Text + '-' + EndY.Text);

{Position the file pointer for carving the data}
FileSeek(OutputFileHandle, 0, 2);

{carve data from the start point to end point on the y-axis of graph}
for i:=strtoint(StartY.Text) to strtoint(EndY.Text) do
begin

    // Set file position
    //CarveOffset := ((i) * (SECTOR_SIZE div 2)*512);
    CarveOffset := ((i) * count;
    Seek(ExtractFileTwoHandle, CarveOffset );

    // Read data

```

```
BlockRead(ExtractFileTwoHandle, BufXcarve, count, BytesRead);

//write carved data to output file
FileWrite(OutputFileHandle, BufXcarve, BytesRead);

end;

//Close output file handle
FileClose(OutputFileHandle);

//Close input files
closefile(ExtractFileHandle);
closefile(ExtractFileTwoHandle);
end;

end.
```

