Projection-based Perceptual Video Hashing

A

thesis submitted

for the award of the degree of

DOCTOR OF PHILOSOPHY

by

SANDEEP R.

Department of Electronics and Electrical Engineering

Indian Institute of Technology Guwahati

Guwahati - 781039, Assam, India

2019

DEDICATION

"ಕಾಯಕವೇ ಕೈಲಾಸ" - ಬಸವಣ್ಣ

"ಇರುವುದೆಲ್ಲವ ಬಿಟ್ಟು ಇರದುದರೆಡೆಗೆ ತುಡಿವುದೆ ಜೀವನ" - ಗೋಪಾಲಕೃಷ್ಣ ಅಡಿಗ

"Work is Worship" - Lord Basavanna
"Life is leaving everything we have and craving for things which we do not have" - Poet Gopalakrishna Adiga

I would like to dedicate this thesis to my parents, my wife, my son, my teachers, my family, my students and my friends.

DECLARATION

This is to certify that the thesis entitled "**Projection-based Perceptual Video Hashing**", submitted by me to the *Indian Institute of Technology Guwahati*, for the award of the degree of Doctor of Philosophy, is a bonafide work carried out by me under the supervision of Prof. P. K. Bora. The content of this thesis, in full or in parts, have not been submitted to any other University or Institute for the award of any degree or diploma.

Dated:
Guwahati

Sandeep R.
Dept. of Electronics and Electrical Engg.,
Indian Institute of Technology Guwahati,
Guwahati-781039, Assam, India.

CERTIFICATE

This is to certify that the thesis entitled "**Projection-based Perceptual Video Hashing**", submitted by Sandeep R. (10610228), a research scholar in the *Department of Electronics and Electrical Engineering*, *Indian Institute of Technology Guwahati*, for the award of the degree of Doctor of Philosophy, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in my opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dated:
Guwahati

Dr. Prabin Kumar Bora
Professor
Dept. of Electronics and Electrical Engg.,
Indian Institute of Technology Guwahati,
Guwahati-781039, Assam, India.

ACKNOWLEDGMENTS

The work would not have been completed without the contribution and support of many people. Here, I would take the opportunity to thank all of them.

First and foremost, I would like to express my heartfelt gratitude to one of the most wonderful human-being, technically sound person, my supervisor, Prof. Prabin Kumar Bora for his support, guidance and encouragement. I am indebted to him for the time and the patience he has shown in carefully correcting my manuscripts and training me to become a researcher with principles.

My heartfelt thanks to my doctoral committee members, Prof. Kannan Karthik, Prof. S. V. Rao and Prof. Rajesh A. for their moral support, thorough evaluations and suggestions that improved my research work.

My sincere thanks to Prof. Ashok Rao, Former Head, Network Project at CEDT, IISc Bengaluru for helping me in correcting my thesis.

I am also thankful to the Head of the Department and the other faculty members for their kind help in carrying out this work. I express my gratitude to all the members of the research and technical staff of the Department for their timely help.

It was wonderful to meet and have a great group of friends. They would be available at all times for any help. A big thanks to the following friends/students for their support they showered upon me: Dr. Tousif Khan N., Mr. Jyoti Prakash Medhi, Dr. Gaurav Yadav, Dr. Atul Kumar, Mr. Kiran Francis, Mr. Harshit B., Mr. Rahul Sanal, Mr. Ganji Sreeram, Mr. Mrinmoy Bharadwaj, Mr. K. Balaji Rao, Dr. Govindaraj P., Dr. C. M. Vikram, Mr. P. Dileep, Mr. Aniruddha Mazumdhar, Dr. Brijesh Kumbhani, Dr. P. Sandeep, Ms. Asha Kiran M. U., Mr. Navaneeth S. Rao and Ms. A. P. Aisiri. I would like to extend my special thanks to Mr. Saksham Sharma and Mr. Mayank Thakur for extending their support in my research work.

I would like to thank Dr. S Padam Priyal and Dr. Gaurav Saxena for providing the thesis template. I would like to acknowledge that I've used the thesis of Dr. Navjit Saikia as a reference for writing my thesis. I would like to thank Dr. M. Li and Dr. V. Monga for providing the MATLAB code of their work.

My deepest gratitude goes to my caring parents, dearest wife, my naughty son and my family for their constant support and love, without whom I could not have completed this work.

Finally, I would like to thank the almighty and all those people who have helped and contributed directly or indirectly in completing this work.

Sandeep R.

ABSTRACT

A perceptual hash function for a video generates a fixed-length binary string called the perceptual hash on the basis of the perceptual contents of the video. This hash must be robust to the manipulations that preserve the perceptual contents of the video and fragile to the modifications that vary the perceptual contents of the video. Developing a perceptual video hashing method satisfying the conflicting properties is a challenge. This thesis generates the perceptual hash from the video based on the projection onto a sub-space by utilizing both the spatial and temporal properties of the video.

This thesis first generates a perceptual hash from the video using the 3D-radial projection of the pixels and assesses the differentiating capabilities and the perceptual robustness of the hash generated. This method is the 3D extension of the well established 2D radial projection-based image hashing. The pixel luminance values of randomly located sub-cubes are radially projected to calculate the variance of the pixels along the radial lines. The hash is obtained in two different ways. In Scheme-I, the variances along the radial lines are averaged, and then projected onto the discrete cosine transform (DCT) basis to form a compact hash vector. This hash vector is binarized using the median-based quantization. In Scheme-II, the variances along the lines of each sub-cube are projected on the 2D DCT basis, and then averaged to form a compact hash vector. This hash vector is also binarized using the median-based quantization. The performance of this algorithm is assessed using the receiver operating characteristic (ROC) curves. Simulation results indicate that the method performs well against most of the typical content-preserving distortions but poorly against the addition of noise. It also performs well against the malicious attacks.

The thesis proposes to use random projection for the generation of perceptual hash from the video. This method is the 3D extension of the well established 2D random projection-based image hashing. The video is projected onto Achlioptas's random basis. This basis functions are written in the form of a matrix called as Achlioptas's random matrix (ARM), with each element taking +1 and -1 with an equal probability. Hence, the projection involves only addition and subtraction operation. Simulation results show that this perceptual hash is robust to common signal and image processing attacks, but has the scope for improvement. To improve the performance of the perceptual hashing function, the temporal *discrete wavelet transform* (DWT), referred to as the *temporal wavelet transform* (TWT), is used for generating the *temporally informative representative frames* (TIRFs) by extracting the features in the temporal direction. The low-frequency components of the resultant transform domain data are then projected onto

the Achlioptas's random basis to generate the hash. The TWT-ARM based perceptual video hashing function not only reduces the dimensions but also retains the essential features. Simulation results show that the TWT-ARM based video hashing algorithm performs better against the content-preserving attacks when compared to that of the existing video hashing algorithms. The drawback of this algorithm is that the computational complexity increases as the number of video frame increases.

Finally, the thesis exploits the multi-linear subspace projection for the extraction of the perceptual hash from the video. In the literature, the reduced rank PARAllel FACtor (PARAFAC) decomposition, has been successfully applied to extract the perceptual hash for the videos. We propose a robust perceptual video hash algorithm based on a superior tensor decomposition known as the Tucker decomposition. We also propose a method to find the number of components in the factor matrices of the Tucker decomposition. The Tucker decomposition based video hashing algorithm shows superior performance against the most of the image processing attacks with the added advantage of computational efficiency.

An application for indexing and retrieval of near-duplicate videos (NDVs) is developed using the proposed video hashing methods. The performance is evaluated using average precision-recall curves. The experimental results on a moderate-size video database shows that the Tucker decomposition and the TWT-ARM based video hashing algorithms perform better in retrieving the NDVs.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

2D	Two Dimension
3D	Three Dimension
3D-DWT	Three Dimensional Discrete Wavelet Transform
3D-RASH	Three Dimensional Radial Projection
ALS	Alternating Least Squares (ALS)
ARM	Achlioptas's Random Matrix
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CGO	Centroid of Gradient Orientation
CR	Compression Ratio
CGO	Centroid of Gradient Orientations
DCT	Discrete Cosine Transform
FJLT	Fast Johnson-Lindenstrauss transform
fps	frames per second
GoF	Group of Frames
HD	Hamming Distance
HOOI	Higher-order Orthogonal Iteration
HOSVD	Higher-order Singular Value Decomposition
JL	Johnson-Lindenstrauss
LL	Low pass band
LRTA	Low Rank Tensor Approximation

MPEG	Moving Pictures Expert Group
NDV	Near-Duplicate Video
NDVD	Near-Duplicate Video Detection
NDVR	Near-Duplicate Video Retrieval
NHD	Normalized Hamming Distance
NMF	Non-negative Matrix Factorization
PARAFAC	PARAllel FACtor
PCA	Principal Component Analysis
PVHA	Perceptual Video Hashing Algorithm
RASH	Radial hASH
RBT	Random Bases Transform
RAV	RAdial Variance
RGB	Red Green Blue
ROC	Receiver Operating Characteristic
SIFT	Scale Invariant Feature Transform
STIRIs	Spatio-Temporally Informative Images
SVD	Singular Value Decomposition
TIRIs	Temporally Informative Representative Images
TIRFs	Temporally Informative Representative Frames
TWT	Temporal Wavelet Transform

Chapter 1

Introduction

Contents of the Chapter

The rapid pace of growth in the field of VLSI technology and the digital communication has led to an unprecedented increase in the volume of videos being produced, stored, broadcast, browsed or viewed every day via digital devices. In addition to this, the easy availability of a number of software tools to access and modify the video has led to an increase in digital forgery and the unauthorized use of the video, to the extent that makes the integrity verification and content identification of the video very necessary, demanding and challenging.

One possible solution to secure the video data is to use the conventional cryptographic techniques. But these techniques take a long time to directly encrypt the video data, as the data size is usually huge. Further, the traditional cryptosystems are extremely sensitive to the input data. Even one-bit change in the input data changes the cryptosystem output dramatically. However, this is not a necessary requirement for video data because of the characteristics of human perception. The original video and the one that has undergone content-preserving manipulations are perceived same even though their digital representations may be different. Thus, the cryptographic technique fails to authenticate the video and to search the video in an extensive database. Watermarking techniques can be used for securing the video data and video authentication [1]. The drawback of a watermarking method is that the video is degraded

Table 1.1: Notations involved in defining the desirable properties of perceptual hash function.

Notation	Description
$\mathcal{V}$	Space of finite videos
$\mathcal{K}$	Space of secret keys
V	Input video
K	Secret key
k	Length of the hash
$\mathbf{h}_v$	Hash vector for the video
$H(V,K)$	Perceptual hashing function
V_{sim}	Video perceptually similar to V
V_{diff}	Video perceptually different from V
$d_{NH}(\cdot,\cdot)$	Normalized Hamming distance

after the watermark is embedded. One solution to address this problem is to use the perceptual video hashing technique. A perceptual video hash function maps the video data to a short binary string based on its appearance to the human eye. The hash values obtained by applying the hash function to perceptually similar video data should remain the same. For perceptually distinct video data, the hash function should produce different hash values.

The flow of the chapter is as per the following. The definition and the detailed discussion on the properties of the perceptual video hashing function are presented in Section 1.1. The applications of perceptual video hash are enumerated in Section 1.2. A brief literature review on perceptual video hashing algorithms is discussed in Section 1.3. The motivation and problem definition are presented in Section 1.4. The contributions of the thesis are summarized in Section 1.5. The organization of the thesis is outlined in Section 1.6.

1.1 Definition and Properties of Perceptual Video Hashing Function

Definition: A *perceptual video hashing function* extracts a compact and a fixed-length binary string on the basis of the perceptual content of the video. The hash extraction process usually involves a secret key to include security. The hash should be *robust* to the *content-preserving manipulations* and *sensitive* to the *content-changing manipulations*.

Properties: Let $\mathbf{h}_v = H(V,K)$ represent the hash of a video V using a secret key K. The notations used for defining the desirable properties of a perceptual hash function are given in Table 1.1. The desirable properties of $\mathbf{h}_v$ are as follows [2, 3, 4, 5, 6]

1. **Non-invertibility**: It should be practically impossible to deduce the content of V from $\mathbf{h}_v$. This is denoted as

$$\mathbf{h}_v \nrightarrow V \tag{1.1}$$

where $\nrightarrow$ denotes non-invertibility.

2. **Conciseness**: The size of the hash should be compact in nature, thus

$$Size\left(\mathbf{h}_v\right) \ll Size\left(V\right) \tag{1.2}$$

3. **Visual robustness**: V and V_{sim} should have similar hash values i.e., the normalized Hamming distance (NHD) between the hashes of perceptually similar videos using the same secret key should be very small. Mathematically, it can be shown as

$$E\left[d_{NH}\left(\mathbf{h}_v, \mathbf{h}_{sim}\right)\right] \cong 0 \tag{1.3}$$

where $\mathbf{h}_v$ and $\mathbf{h}_{sim}$ represent the hash value of V and V_{sim} respectively generated using the same key. The NHD is defined by

$$d_{NH} = \frac{d_H}{k}$$

where d_H represent the Hamming distance (HD) between the hashes.

4. **Visual fragility**: V and V_{diff} should have different hash values i.e., the NHD between the hashes of perceptually different videos using the same key should be approximately equal to 0.5. Mathematically,

$$E\left[d_{NH}\left(\mathbf{h}_v, \mathbf{h}_{diff}\right)\right] \cong 0.5 \tag{1.4}$$

where $\mathbf{h}_v$ and $\mathbf{h}_{diff}$ represent the hash value of V and V_{diff} respectively generated using the same key.

5. **Unpredictability**: The hash value is intractable without the secret key i.e., the NHD between two hashes of same video using the different keys should be approximately equal to 0.5. Mathematically,

$$E\left[d_{NH}\left(\mathbf{h}_v, \mathbf{h}^{'}{}_v\right)\right] \cong 0.5 \tag{1.5}$$

where $\mathbf{h}_v$ and $\mathbf{h}^{'}{}_v$ represent the hash value of V generated using two different keys K and $K^{'}$, respectively.

The perceptual video hash need not satisfy all the properties predominantly and simultaneously. It depends upon the application for which the perceptual video hash is being used. For example, consider the perceptual hash is being used for video authentication or integrity verification, then the hash must be sensitive to the minimal malicious modifications made to the video. In this case, the perceptual hash must satisfy the visual fragility property primarily. If the perceptual video hash is being used for copyright protection application, then it must satisfy the unpredictability property predominantly. Finally, if the perceptual hash is being used for the retrieval application, then it must satisfy the robustness property considerably. In Section 1.2, some of the applications of perceptual video hashing is discussed in brief.

Developing a perceptual video hash satisfying all the above properties is quite demanding. This thesis investigates perceptual video hashing techniques and proposes new perceptual video hashing techniques.

1.2 Applications of Perceptual Video Hashing

1. **Content-based identification** [7]: The block diagram for the content-based video identification is shown in Figure 1.1. Content-based identification is the problem of searching for digital videos in large databases. "Content-based" means that the search will analyze the actual content of the video. Developing the content-based video identification application involve two phases: (i) database creation phase; and (ii) content identification phase. In the database creation phase, the perceptual hash is extracted for the videos and stored in the database along with the meta-data of the video such as the video title and owner of the video.

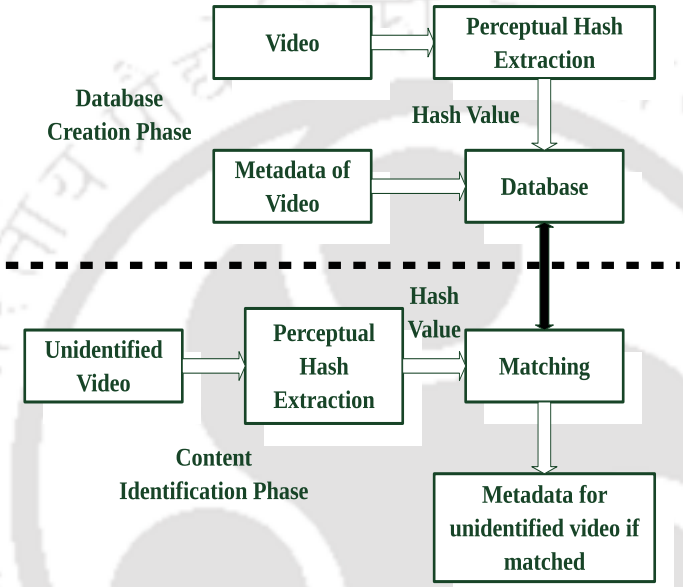


Figure 1.1: Content-based video identification.

Let, $\mathbf{h}_j,\ j = 1, 2, \ldots, N$ be the hash vectors for V_j videos generated using the perceptual video hashing function, respectively. These V_j videos and the corresponding hashes $\mathbf{h}_j$ along with the meta-data are stored in the database. In the identification phase, the hash $\mathbf{h}_{query}$ is computed for the query video V_{query} using the same perceptual video hashing function, and then compared with all the hashes in the database using a distance metric. If $d(\mathbf{h}_j, \mathbf{h}_{query}) \leq T_h$ then V_j and V_{query} are similar otherwise they are dissimilar, where d is a suitable distance metric and T_h is a suitably chosen threshold. Once $\mathbf{h}_{query}$ and $\mathbf{h}_j$ are found to be similar, the corresponding video and the meta-data are identified.

2. **Integrity verification**: The integrity verification problem refers to the confirmation of the video content being authenticated and not maliciously modified. The verification can be done using one of the following two methods, namely

   (i) **Watermarking scheme**: A watermarking scheme involves two phases as shown in Figure 1.2.

   (a) **Creation of watermarked video**: The perceptual hash, $\mathbf{h}_v$ is generated from the video, V using the secret key, K. Both V and $\mathbf{h}_v$ is applied to the watermark

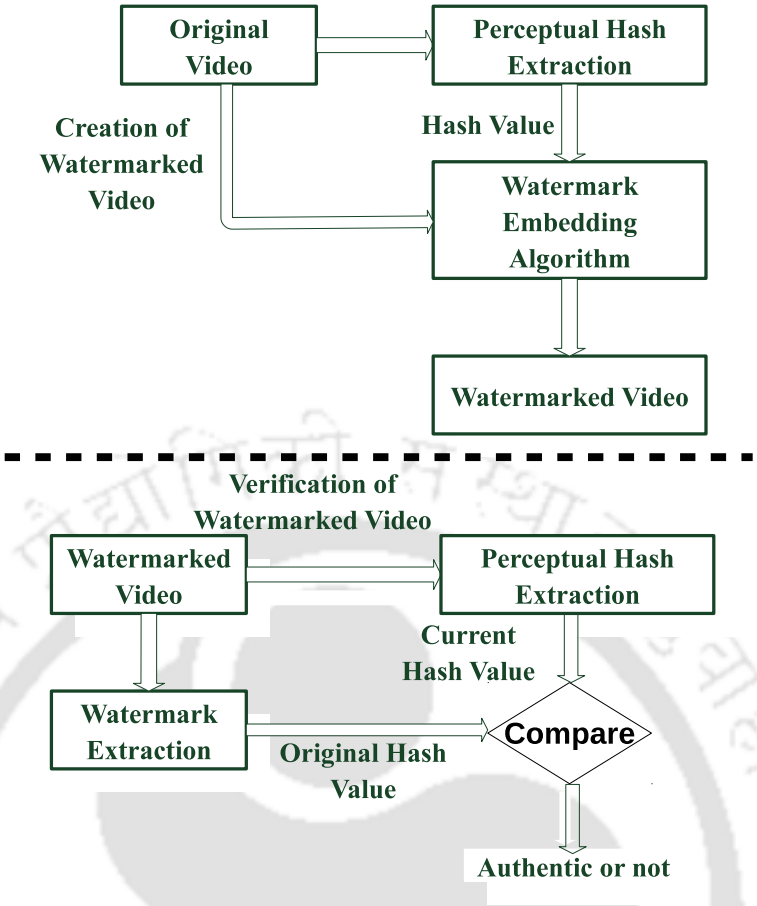


Figure 1.2: Video integrity verification using watermarking.

embedding algorithm to obtain the watermarked video, V_{wm}.

(b) **Verification of watermarked video**: The perceptual hash $\mathbf{h}'_v$ is extracted from the received (query) watermarked video V'_{wm} using K. Simultaneously, the watermark is extracted from V'_{wm} and then the $\mathbf{h}_v$ is computed. Both the hashes $\mathbf{h}'_v$ and $\mathbf{h}_v$ are matched for similarity using a distance metric. If $d\left(\mathbf{h}_v, \mathbf{h}'_v\right) \leq T_h$ then V'_{wm} is authentic otherwise V'_{wm} is not authentic, where d is a suitable distance metric and T_h is a suitably chosen threshold.

The following are the drawbacks of the watermarking technique:

- The watermark can easily be removed or cropped
- It degrades the quality of the video as the watermark is a distracting element
- It is a time-consuming technique

(ii) **Digital signature scheme** [8]: This scheme involves two phases as shown in Figure 1.3.

(a) **Digital signature creation phase**: The perceptual hash vector $\mathbf{h}_v$ is extracted from the video, V for which the digital signature is to be generated using a secret key K. $\mathbf{h}_v$ and K are encrypted using the private key K_{pr}. The output of encryption is the digital signature, V_{ds} for V. The input video V and the corresponding digital signature V_{ds} is sent using a secured channel.

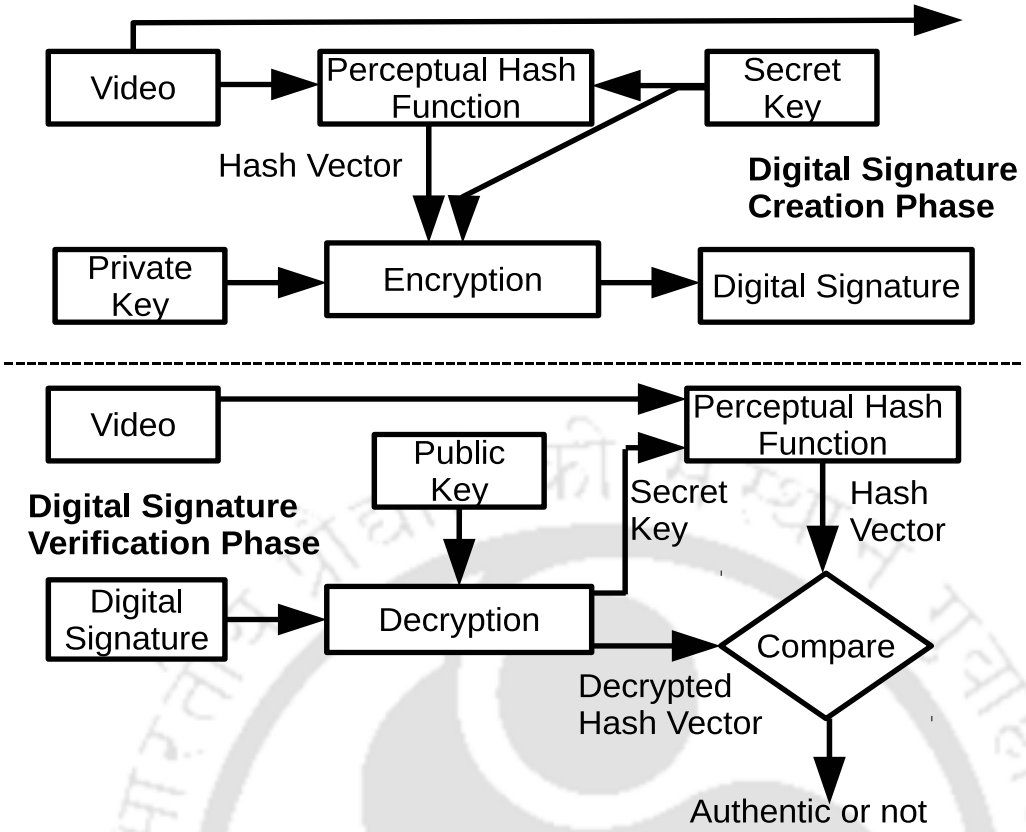


Figure 1.3: Video integrity verification using digital signature.

(b) **Digital signature verification phase**: Using a public key K_{pu} the received digital signature $V^{'}{}_{ds}$ is decrypted to generate the hash vector $\mathbf{h}_v$ and the secret key K. Using this K, the perceptual hash $\mathbf{h}^{'}{}_v$ is extracted from the received video (query) V_{query} for which the authenticity is to be verified. $\mathbf{h}_v$ and $\mathbf{h}^{'}{}_v$ is compared using a distance metric. If $d\left(\mathbf{h}_v, \mathbf{h}^{'}{}_v\right) \leq T_h$ then V_{query} is authentic otherwise V_{query} is not authentic, where d is a suitable distance metric and T_h is a suitably chosen threshold.

Even though this scheme has the disadvantage of requiring additional overhead bits and bandwidth, it is chosen over the watermarking technique due to the following advantages [9]:

- The video data is not modified
- All of the existing videos can be digitally signed
- The scheme can be used in authentication systems based on public-key

3. **Near-duplicate video retrieval**: The block diagram of the retrieval of near-duplicate videos (NDVs) application, based on perceptual video hashing is shown in Figure 1.4. This application retrieves the NDVs rather than relevant videos. This scheme involves two phases. They are,

(i) **Database creation phase**: In this phase, hashes are extracted from a large set of videos. Let, $\mathbf{h}_j$, $j = 1, 2, \ldots, N$ be the hash vectors for V_j videos generated

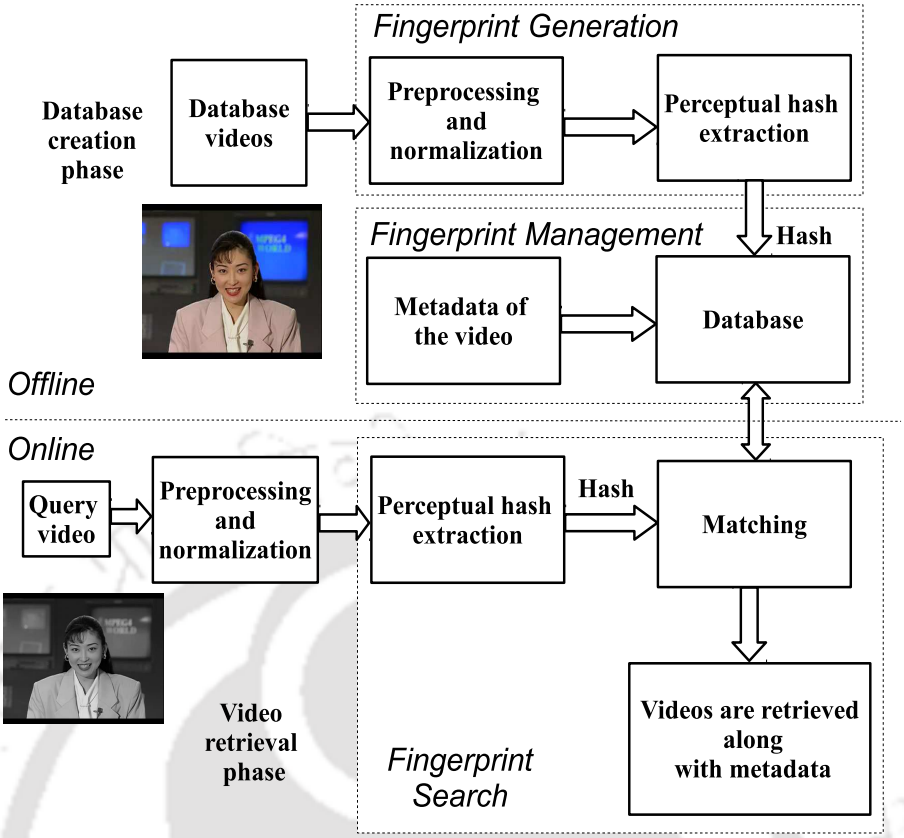


Figure 1.4: Content-based near-duplicate video retrieval.

using the perceptual video hashing algorithm, respectively. The V_j videos and the corresponding hashes $\mathbf{h}_j$ along with the meta-data are stored in the database.

(ii) **Video retrieval phase**: In the retrieval phase, the hash $\mathbf{h}_{query}$ is computed for the query video V_{query} using the same perceptual video hashing algorithm, and then compared with all the hashes in the database using a distance metric. If $d(\mathbf{h}_j, \mathbf{h}_{query}) \leq T_h$ then all the V_js (NDVs) are retrieved, where $j = 1, 2, \ldots N$, d is a suitable distance metric and T_h is a suitably chosen threshold.

1.3 Literature Review on Perceptual Video Hashing

Many perceptual video hashing algorithms have been proposed in the literature. This section presents a brief overview of these. The objective of this section is three-fold. (i) we review some of the perceptual video hashing algorithms and classify the algorithms based on a criterion. (ii) we examine the perceptual video hashing algorithms and their performance in terms of robustness and security. (iii) we establish the scope for the current research.

The methodologies existing in the literature for the generation of perceptual video hash can be classified into *projection-based* and *non-projection based*. Many of the perceptual hashing algorithms generate a compact hash function using the projection of the image/video data onto a subspace. Such subspace projections available in the literature can be further classified into two types: *deterministic* [10], [11], [12], [13], [14], [15] and *random projections* [16, 17]. Some of the deterministic projections are the Discrete Cosine Transform (DCT), Discrete Wavelet

Transform (DWT) and tensor factorization [18], [19]. Important random projections are the normal distribution matrix, the Achlioptas's Random Matrix (ARM) [20], [21] and the Fast Johnson- Lindenstrauss transform (FJLT) [22], [23]. The perceptual video hashing algorithms based on random projection of entire video is still an unexplored area. However, for the sake of completeness perceptual image hashing based on random projection is discussed in Subsection 1.3.3. The non-projection-based perceptual video hashing algorithms involve histogram [24], local and global features of the frame [25].

1.3.1 Non-projection-based Perceptual Video Hashing Algorithms

A brief description of non-projection based perceptual video hashing algorithms is presented below.

Histogram based Perceptual Video Hashing Algorithm

The authors in [24] have generated the perceptual hash based on the histogram of the frames by considering the videos having minimal variation in the temporal direction. The brief description of the algorithm is as follows.

(i) The video is decoded into individual frames.

(ii) Each colour frame is converted into grey frames.

(iii) Frames are temporally averaged.

(iv) A Gaussian filter is used to remove noise and subsequently bilinear resizing is done.

(v) The resulting image is quantized by obtaining threshold values using a cumulative histogram of the image to obtain the binary hash.

The algorithm is robust against many signal processing attacks and can distinguish original and attacked videos in case of minimally changing videos.

Statistics and Harris Detector based Perceptual Video Hashing Algorithm

The authors in [25] have generated the perceptual hash by using the statistical features and the Harris detector on the video frames. The steps of the algorithm are enumerated below.

(i) The input color video of arbitrary spatial size is converted to a grey video of predefined size and then filtered to remove any noise. Finally, the frames are enhanced by applying histogram equalization.

(ii) Statistics such as the standard deviation, a third-order cumulant and a fourth-order cumulant is applied on the overlapping sub-blocks of the frame to generate an intermediate hash that captures the global frame features. The Harries detector is applied on the preprocessed and normalized frame to generate the second intermediate hash that captures the local frame features. Both the intermediate hashes are concatenated to form a frame hash.

(iii) This frame hash is obtained for every i^{th} frame. All the frame hashes are added and then mean-based quantization is applied to generate a compact hash vector for the video.

This algorithm is robust against compression, sharpening, and slight geometric distortion, such as rotation and cropping.

Both the algorithms discussed above are implemented on a small database and robust only to slight content-preserving distortions.

1.3.2 Deterministic Projection-based Perceptual Video Hashing Algorithms

Most of the perceptual video hashing algorithms existing in the literature are based on deterministic projection. Some of popular deterministic projection-based perceptual video hashing algorithms are discussed below.

Radial Projection based Video Hashing Algorithm

180 lines with a separation of 1° in their angular orientation and each passing through the centre of the image are chosen. The variance of the luminance of the pixels on each line is computed. These 180 variances form the feature vector. The authors in [11] argue that this feature vector provides the best trade-off in terms of robustness and discriminating capabilities. Suppose $I(x,y)$ denotes the luminance value of a pixel (x,y) and N_ϕ denotes the number of pixels on the line at an angle ϕ. Then, the projected feature vector is given by

$$\mathbf{P} = \begin{bmatrix} P(1^\circ) & P(2^\circ) & \dots & P(180^\circ) \end{bmatrix}^T$$

where

$$P(\phi) = \frac{\sum\limits_{(x,y)\in N_\phi} I^2(x,y)}{N_\phi} - \left(\frac{\sum\limits_{(x,y)\in N_\phi} I(x,y)}{N_\phi} \right)^2, \quad \phi = 1^\circ, 2^\circ, \dots, 180^\circ \tag{1.6}$$

The $\mathbf{P}$ vector is centred around the origin and normalized with respect to the mean μ and standard deviation σ respectively, to obtain the radial variance (RAV) feature vector,

$$\mathbf{R} = \begin{bmatrix} R(1^\circ) & R(2^\circ) & \dots & R(180^\circ) \end{bmatrix}^T$$

where $R(\phi)$ is given by

$$R(\phi) = \frac{(P(\phi) - \mu)}{\sigma}, \quad \phi = 1^\circ, 2^\circ, \dots, 180^\circ \tag{1.7}$$

The 40 low-frequency coefficients of 180-point DCT of the RAV feature vector are chosen to obtain a compact hash vector. Mathematically, the radial hash (RASH) vector

$$\mathbf{D} = \begin{bmatrix} D(1^\circ) & D(2^\circ) & \dots & D(180^\circ) \end{bmatrix}^T$$

where $D(n)$ is given by,

$$D(n) = \sqrt{\frac{2}{180}} \times \sum_{\phi=1^\circ}^{180^\circ} \left(R(\phi) \times \cos\left(\frac{\pi \times (2\phi + 1) \times n}{2 \times 180} \right) \right), \quad n = 1, 2, \ldots, 40 \tag{1.8}$$

Finally, each of the 40 low-frequency DCT coefficients is quantized to 8 bits to obtain a 320-bit image hash.

The extension of the image hashing algorithm to video hashing [10] is as follows. The video sequence is modelled as a collection of video shots, and key frames are selected from those video shots. A shot is identified by selecting the boundary frames. If the ℓ_1 distance between the 64-bin luminance histograms of the current frame and the previous frame is greater than a particular threshold, then the current frame is considered as the boundary frame. Within the frames of a shot, the disparity between the frames is measured [26]. A frame is selected as the keyframe if it minimizes the disparity measurement with its preceding frame. Finally, the hash value is obtained by applying the image hashing algorithm on these keyframes.

The radial projection technique, as proposed in [10], is dependent on keyframes corresponding to a video. This was a major limitation as this technique is exposed to all the non-keyframe-based attacks such as frame dropping, malicious attacks on non-keyframes, and unwanted frame insertions.

3D-DCT based Video Hashing Algorithm

Coskun and Sankur in [12, 13], have proposed a video hashing technique based on the 3D subspace projections of the luminance components of a video on a DCT basis. The projected low-frequency 3D-DCT coefficients of the normalized video are quantized to obtain the hash. Following are the steps of the algorithm:

(i) Normalization: Normalize the input video clips via temporal sub-sampling and spatial resizing. This standardization of videos to $V_{norm}(W, H, F) \in \mathbb{R}^{W \times H \times F}$, makes the hashes robust to various spatial dimensions and frame rates of the video.

(ii) Transformation: 3D-DCT is applied to $V_{norm}(W, H, F)$ to obtain 3D array of DCT coefficients $V_{DCT}(W, H, F)$.

(iii) Coefficient selection: 64 coefficients in the form of $4 \times 4 \times 4$ cube in the low pass band is selected for the generation of hash.

These selected coefficients are robust against various signal processing attacks. The 3D-DCT based video hash is not secure as the generation of hash is not dependent on any secret key. To include randomness, the authors have generated DCT bases of dimensions $W \times H \times F$ having random frequencies with the help of a secret key. These 3D Random Bases Transform (3D-RBT) bases are then passed through a low-pass filter in all three dimensions to remove the unwanted high-frequency random basis functions, that reduces the robustness of the hash function. The RBT is obtained via the projection of the 3D video data onto these 3D basis functions. Finally, after the 3D transform (DCT/RBT) is applied, the selected transform coefficients are binarized using the median as the threshold. The perceptual video hash based on 3D randomized basis showed inferior performance compared to that of the deterministic basis.

Centroid of Gradient Orientations based Video Hashing Algorithm

Sunil and Yoo in [7, 27] have obtained the hash values based on the *Centroid of Gradient Orientations*(CGO). The CGO values give the direction (angle) information from the difference of adjacent pixel values. Each frame of the normalized video was divided into blocks and then for each block, the CGO was calculated. The final hash for the video was obtained by concatenating the CGOs of all blocks. The steps of the algorithm are as follows:

(i) Normalization: The input videos are temporally sampled to K number of frames and converted to grey videos. Next, videos are spatially re-sized to a frame size of $X \times Y$.

(ii) Frame division: Each re-sized frame is divided into $N \times M$ blocks.

(iii) Centroid of gradient orientation calculation: Let $B_{n,m,k}$ be the block in the n^{th} row and m^{th} column of the k^{th} frame and $c\,[n,m,k]$ be the centroid of the block $B_{n,m,k}$, $(1 \leq n \leq N, 1 \leq m \leq M)$. Then centroid of gradient orientation for each block $B_{n,m,k}$ is given by

$$c\,[n,m,k] = \frac{\sum\limits_{(x,y)\in B_{n,m,k}} r\,[x,y,k]\,\theta\,[x,y,k]}{\sum\limits_{(x,y)\in B_{n,m,k}} r\,[x,y,k]} \tag{1.9}$$

where

$r\,[x,y,k] = \sqrt{G_x^2 + G_y^2}$ is the gradient magnitude,

$\theta\,[x,y,k] = \tan^{-1}\left(\frac{G_y}{G_x}\right)$ is the gradient direction,

$G_x = f\,[x+1,y,k] - f\,[x-1,y,k]$ and $G_y = f\,[x,y+1,k] - f\,[x,y-1,k]$ are the approximations of gradient in $x-$direction and $y-$direction respectively and

$f\,(x,y,k)$ is the luminance value at the location (x,y) in the k^{th} frame.

(iv) Hash computation: The hash vector $\mathbf{c}_k$ of the k^{th} frame is obtained by

$$\mathbf{c}_k = [c\,[1,1,k]\,, c\,[1,2,k]\,, \ldots, c\,[N,M,k]] \tag{1.10}$$

This fingerprint has information about the distribution of edges in each video frame. The final hash vector for a video is obtained by concatenating all the $\mathbf{c}_k$s; $1 \leq k \leq K$.

In this method, the gradients are based on the pixel differences and hence robust against global changes in pixel intensities such as brightness, colour, and contrast. The algorithm is robust to most perceptually insignificant operations but weak under geometric distortions like the rotation attack.

Temporally Informative Representative Images (TIRI) based Video Hashing Algorithm

In [28, 29], Mani et al., have generated the video hash based on temporally informative representative images (TIRIs). The TIRIs are generated using weighted superposition of luminance values along the temporal direction of a predefined number of frames chosen from a given video.

The image hashes are generated from the individual TIRIs and then, concatenated to generate the video hash. The steps of the algorithm are as follows.

(i) The input video is normalized to $W \times H_h$ and divided into shots by putting L frames in a particular time range together.

(ii) The TIRIs are generated by

$$a_{x,y} = \sum_{l=1}^{L} w_l lu_{x,y,l} \tag{1.11}$$

$w_l = \gamma^l$ = exponential weighting

$lu_{x,y,l}$ = luminance value of $(x,y)^{\text{th}}$ pixel in l^{th} frame and L number of frames in the shot.

(iii) The 2D-DCT of the TIRI frames is calculated by

$$c_{p,q} = 2^{1-\frac{\delta_p+\delta_q}{2}} \sum_{x=0}^{W-1} \sum_{y=0}^{H_h-1} a_{x,y} \cos(\alpha_x p) \cos(\beta_y q) \tag{1.12}$$

where $\alpha_x = \frac{\pi(2x+1)}{2W}$, $\beta_y = \frac{\pi(2y+1)}{2H_h}$, $\delta_p = \begin{cases} 1; & p = 0 \\ 0; & \text{elsewhere} \end{cases}$ and $\delta_q = \begin{cases} 1; & q = 0 \\ 0; & \text{elsewhere} \end{cases}$

(iv) $M_b \times M_b$ block of coefficients of $c_{p,q}$ are selected and vectorized. Median-based quantization is applied on this vector to generate the perceptual video hash.

The resulting hashes were robust to a range of attacks including noise addition, rotation, time shift and frame dropping but not for the malicious attacks.

Perceptual Video Hashing Algorithm for Scalable Video

In [30], [31], the authors proposed the perceptual video hashing algorithm to extract the hashes for scalable video using the 3D- discrete wavelet transform (DWT). They have computed the video hash at a group-of-frames(GoFs) level from the spatiotemporal low-pass bands of the wavelet transformed GoFs. Following are the steps of the algorithm:

(i) The video is grouped into GoFs with 8, 16 or 32 frames and a GoF is denoted as G'.

(ii) The 3D-DWT is applied on the GoFs. The first-level temporal decomposition of the GoF results in one temporal low-pass band and one temporal high-pass band. This temporal low-pass band is recursively decomposed to obtain one temporal low-pass band and multiple temporal high-pass bands at the highest level of decomposition. The temporal low-pass band is further decomposed into multiple spatial low-pass and high-pass bands. Thus, the 3D-DWT on the GoFs results in one spatiotemporal low-pass (LLL) bands at the highest levels of the temporal and spatial decompositions. The LLL bands act as the spatiotemporally informative images (STIRIs).

(iii) The perceptual hash for G' is obtained from the LLL band of the 3D-DWT decomposition using one of the two variants. In the first variant, the LLL band is divided into perceptual blocks, and then, the coefficients are binarized to obtain the perceptual hash for G'. This variant results in a large hash size that has weak visual fragility and unpredictability properties. In the second variant, a compact hash is obtained by binarizing the forward and backward cumulative averages of the local means of the perceptual blocks in the LLL band.

The performance of the video hashing algorithms was evaluated on a small database of 14 videos. The algorithm is not robust towards geometric transformation.

Low Rank Tensor Approximation (LRTA) based Video Hashing Algorithm

In [14, 15], Li and Monga have modelled videos as third order tensors. The authors experimentally validate that this type of video modelling captures its spatial and temporal essence. The multi-linear sub-space projections of tensors, such as low-rank tensor approximations (LRTAs) via PARAllel FACtor analysis (PARAFAC) [18] are used for the generation of perceptual hash from the videos. The authors perform a random 3D tiling of the video via overlapping sub-blocks and compute LRTAs from these sub-blocks. Each sub-block treated as a third order tensor was approximated using a rank one tensor. In other words, each third order tensor is factorized into the outer product of three vectors. These vectors are used to form the perceptual hash. The steps of the algorithm are enumerated below.

(i) Normalization: Though the temporal sub-sampling and the spatial re-sizing, each input video $\mathcal{V}$ is normalized to a size of $I \times J \times K$ and then converted to a sequence of grey-level frames of video.

(ii) Randomization: Based on a secret key K, Q randomly overlapping sub-cubes $\mathcal{V}_i \in \mathbb{R}^{M \times N \times P}$, $i = 1, \ldots, Q$, $M < I$, $N < J$, $P < K$ are selected such that they cover the entire video $\mathcal{V}$ approximately.

(iii) Tensor factorization: Next rank-r PARAFAC tensor factorization is performed on each sub-tensor. This results in three sets of vectors for each sub-tensor where the two sets of vectors $\mathbf{x}_i \in \mathbb{R}^{M.r}$, and $\mathbf{y}_i \in \mathbb{R}^{N.r}$ belong to the spatial dimensions and the third vector $\mathbf{z}_i \in \mathbb{R}^{P.r}$ belong to the temporal dimension for $i = 1, \ldots, Q$. A rank $r = 1$ is used for tensor approximations.

(iv) Hash computation: The hash vector $\mathbf{h}_K \in \mathbb{R}^{(M+N+P)}$ is obtained through arithmetic averaging given by:

$$\mathbf{h}_K = \left[\frac{\sum_{i=1}^{Q} \mathbf{x}_i}{Q}, \frac{\sum_{i=1}^{Q} \mathbf{y}_i}{Q}, \frac{\sum_{i=1}^{Q} \mathbf{z}_i}{Q} \right] \tag{1.13}$$

Here, the spatial components of the LRTA are unaffected when the attack is temporal, and the temporal components are unchanged when the attack is spatial. The performance of the

algorithm was found to be good to most of the content-preserving distortions. Further, there is a scope for the generalization of tensor decomposition to improve the performance of the algorithm.

1.3.3 Random Projection based Perceptual Image Hashing

In random projection, when the original high-dimensional space is projected onto a low-dimensional space using a random projection matrix, the distance between the vectors are preserved, provided it satisfies the Johnson-Lindenstrauss (JL) lemma [32]. Alion and Chazelle introduced the FJLT for reducing a high-dimensional data into a lower-dimension [22, 23]. The FJLT maps n points from $\mathbb{R}^d$ to $\mathbb{R}^k$ with ϵ distortion, where the low-dimension k is given by $k = \mathrm{c}'\epsilon^{-2}\ln n$ and c' is a constant. A random embedding $\mathbf{R}_{FJLT} \sim \mathrm{FJLT}\,(n, d, \epsilon)$ can be obtained as a product of real-valued matrices

$$\mathbf{R}_{FJLT} = \mathbf{PTD} \tag{1.14}$$

where $\mathbf{T}$ is the $d \times d$ normalized Hadamard matrix, and $\mathbf{P}$ and $\mathbf{D}$ are $k \times d$ and $d \times d$ matrices pseudo-randomly dependent on the seed. The elements p_{ij} of $\mathbf{P}$ are drawn independently according to the following distribution,

$$p_{ij} = \begin{cases} N\left(0, q^{-1}\right) & \text{with probability } q \\ 0 & \text{with probability } (1-q) \end{cases} \tag{1.15}$$

where $q = \min\left\{\frac{c\ln^2 n}{d}, 1\right\}$ and c a large constant. $\mathbf{D}$ is a diagonal matrix, where each diagonal element d_{ij} is drawn independently from$\{-1, 1\}$ with probability 0.5. The FJLT lemma is given by

FJLT Lemma*: Let* $\mathbf{Q}$ *be an arbitrary subset of* n *points in* $\mathbb{R}^d$*, represented as an* $d \times n$ *matrix* $\mathbf{F}$*. Given* $o < \epsilon < 1$ *and* $\mathbf{R}_{FJLT} \sim FJLT(n, d, \epsilon)$ *the following two events occur with probability at-least* $\frac{2}{3}$

(i) $\forall\, \mathbf{x} \in \mathbf{Q}$, $(1-\epsilon)\, k\, \|\mathbf{x}\|_2 \leq \|\mathbf{R}_{FJLT}\mathbf{x}\|_2 \leq (1+\epsilon)\, k\, \|\mathbf{x}\|_2$.

(ii) The mappings $\mathbf{R}_{FJLT} : \mathbb{R}^d \to \mathbb{R}^k$ *requires* $\mathcal{O}\left(d\ln d + \min\left(d\epsilon^{-2}\ln n,\, \epsilon^{-2}\ln^3 n\right)\right)$ *operations.*

Based on the FJLT, Xudong and Wang [17, 16] proposed an image hashing concept that is robust to a large class of content preserving distortions. The results are comparable to the state-of-the art image hashing algorithms such as image hashing based on non-negative matrix factorization (NMF) [5]. The FJLT based image hashing algorithm is as follows:

1. Random sampling: Given an image, pseudo-randomly select p subimages, $\mathbf{A}_i \in \mathbb{R}^{m\times m}, 1 \leq i \leq p$. Vectorize each subimage $\mathbf{A}_i$ to form $\mathbf{s}_i$, a point in a m^2-dimensional space. The feature matrix is given by: $\mathbf{F} = [\mathbf{s}_1\ \mathbf{s}_2\ \ldots\ \mathbf{s}_p]$, with size $m^2 \times p$.

2. Dimension reduction by FJLT: FJLT $\left(p, m^2, \epsilon\right)$ maps the feature matrix from a high-dimensional space $\mathbb{R}^{m^2}$ to a low-dimensional space $\mathbb{R}^k$ with minor distortion as follows

$$\mathbf{H} = \mathbf{R}_{FJLT}\mathbf{F} \tag{1.16}$$

The lower dimension k is set to be $c'\epsilon^{-2}\ln(p)$ where c' is a constant. The intermediate hash is given by $\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2\ \ldots\ \mathbf{h}_p]$ with size $k \times p$.

3. Ordered random weighting: Generate the pseudo-random weight factors $\{\mathbf{w}_i\}_{i=1}^{p}$ from the uniform distribution $\mathrm{U}(0,1)$ such that each $\mathbf{w}_i$ is in $\mathbb{R}^k$. Sort the elements of $\mathbf{h}_i$ and $\mathbf{w}_i$ in a descending order to make sure that larger component of $\mathbf{w}_i$ will be assigned to larger component of $\mathbf{h}_i$. Calculate the final secure hash as $\mathbf{h} = [\langle \mathbf{h}_1\ \mathbf{w}_1\rangle\ \langle \mathbf{h}_2, \mathbf{w}_2\rangle\ \ldots\ \langle \mathbf{h}_p, \mathbf{w}_p\rangle]$ where $\langle\ \rangle$ denotes the inner product between the vectors.

The algorithm was found to be robust to most of the typical image processing attacks except for the addition of Gaussian noise, gamma correction, and rotation attack. The algorithm has a low computational cost.

1.4 Motivation and Problem Definition

Generating a perceptual hash from a video by concatenating the perceptual image hashes extracted from each frame results in a very long hash. Therefore, developing a perceptual video hashing technique is still a challenging problem. There are few methods that exist in the literature as discussed above for the generation of perceptual video hash by treating the video as a spatio-temporal entity. However, these techniques have scope for improvement. The thesis aims at generating perceptual video hashing functions to solve the following generic problems:

1. The *content-authentication problem*: The incidental distortions such as compression, noise addition, filtering and transcoding, that a video undergoes during distribution are unlikely to change the perceptual content and the video can be considered still authentic. The performance of the existing video hashing algorithms for this task has the scope for improvement.

2. The *near-duplicate identification problem*: The rapid improvement in the digital technology and social media has led to the increase in the production and distribution of the videos. In addition, the easy availability of a number of software tools to access and modify the video has led to unauthorized multiple creation of popular videos in different file formats, spatial sizes and frame rates. The large number of near-duplicate videos also cause a huge burden on the database server. Finding the near-duplicate videos in a large database is a challenging problem and perceptual video hashing may prove useful to tackle the problem.

1.5 Summary of Contributions

The major contributions of the thesis are as follows:

1. The thesis proposes the perceptual video hashing algorithms based on the following projection operations on video.

1 **3D radial projections**: The two-dimensional radial projection on images is extended to video.

2 **Achlioptas's random projection**: The two-dimensional random projection on images using the Achlioptas's random matrix (ARM) is extended to video.

3 **Combination of the temporal wavelet transform (TWT) and the Achlioptas's random projection**: The video is projected onto temporally informative representative frames (TIRFs) using the temporal wavelet transform and Achlioptas's random projection are applied on the TIRFs to develop the hash.

4 **Tucker decomposition**: The PARAFAC algorithm is generalized using the Tucker decomposition to develop a more robust perceptual video hashing function. A method for finding the optimum number of components in the factor matrices of the Tucker decomposition is also proposed.

2. The thesis develops an application of perceptual video hashing for retrieving near-duplicate videos from a database.

1.6 Outline of the Thesis

The rest of the thesis is organized as follows:

- Chapter 2 introduces the framework for the three-dimensional radial projection (3D-RASH) based video hashing method and then proposes the algorithm. The algorithm produces the perceptual hash by projecting the luminance values of the pixels along the radial lines of the overlapping sub-cubes in a video and computing the variance of the projected pixels. It presents the complete details about the set up of the video database for the current research downloaded from REEFVID, XIPH, OPEN-VIDEO and TRECVID databases. Further, it also discusses the procedure used for (i) verifying the desired properties of the perceptual video hashing function; (ii) evaluating and comparing the performance of the 3D-RASH based video hashing algorithm with some of the popular video hashing algorithms using the receiver operating characteristic (ROC) curves.

- Chapter 3 presents the framework for the Achlioptas's random projection based video hashing method and then proposes two perceptual video hashing algorithms. The first algorithm generates the perceptual hash by projecting the pixel luminance of the overlapping sub-cubes on to the Achlioptas's random basis. It provides the experimental results in detail to examine the hashing properties. It evaluates and compares the performance of the Achlioptas's random projection based video hashing algorithm with other video hashing algorithm. The second video hashing algorithm is based on the temporal wavelet transform (TWT) and the Achlioptas's random projection. The second algorithm generates the perceptual hash by projecting the pixels in the overlapping sub-cubes in the temporal low-pass band on to the Achlioptas's random basis. It validates the desirable properties of the hash and evaluates the performance of the second video hashing algorithm and also compares the performance with the other video hashing algorithms using the ROC curves on the database mentioned above.

- Chapter 4 proposes to use the Tucker decomposition, a multi-linear subspace projection technique on the overlapping sub-tensors to generate the perceptual hash from the video. It validates the necessary properties of this hash function. It evaluates and compares the performance of the Tucker decomposition based video hashing algorithm with other video hashing algorithms. Evaluates the comparative performance of the algorithm on the database mentioned above in terms of both the ROC curves and the computational complexity.

- Chapter 5 demonstrates the application of perceptual video hashing in retrieving the NDVs. It discusses the state-of-the-art near-duplicate video retrieval algorithms and then proposes the framework to develop the application using different perceptual video hashing algorithms. It gauges the performance of the retrieval system using the average precision-recall curves on three different databases created from REEFVID, XIPH, OPEN-VIDEO and TRECVID databases.

- Chapter 6 concludes the thesis with a summary of the proposed video hashing methods. A few future works are also identified.

Chapter 2

Perceptual Video Hashing based on Three Dimensional Radial Projection

Contents of the Chapter

One of the popular image hashing technique is based on the radial projection of the image pixels. In this, the luminance values of the pixels in the image were projected radially to obtain a RAdial Variance (RAV) vector [11]. The RAV is transformed to the Discrete Cosine Transform (DCT) domain to utilize the energy compaction property and the decorrelation property of the DCT. By choosing and quantizing the low-frequency DCT coefficients, the image hash was generated and called the RASH (Radial hASH) vector. This method of calculating the perceptual hash is simple. The idea as mentioned above was extended to the video by applying the technique to the keyframes of the video [10]. The performance of this algorithm was found to be good to most of the typical image processing attacks.

Inspired by the simplicity of the work proposed by De Roover et al. in [11, 10], we would like to extend the idea of radial projection based hashing from the two-dimension (2D) to the

three dimensions (3D). In this way, the keyframe dependence of RASH for video [10] can be overcome by considering the entire video. Our objective is to

1. Generalize the robust radial projection based hashing method from 2D to 3D(video); and
2. Study the performance of the proposed generalization in comparison with the keyframe-based RASH for video.

The flow of the chapter is as per the following. A brief review on two-dimensional radial projection is presented in Section 2.1. The framework for the three-dimensional radial projection (3D-RASH) based perceptual video hash is described in Section 2.2. The 3D-RASH based video hashing algorithm is proposed in Section 2.3. The details of the database, description of the attacks considered, validation of perceptual hash properties and the performance evaluation of the 3D-RASH based perceptual hash is presented in Section 2.4.

2.1 Perceptual Video Hashing using 2D Radial Projection

The 2D radial projection technique was discussed in Chapter 1. For the sake of completeness, the same is reproduced here. 180 lines with a separation of 1° in their angular orientation and each passing through the centre of the image are chosen. The variance of the luminance of the pixels on each line is computed. These 180 variances form the feature vector. The authors in [11] argue that this feature vector provides the best trade-off in terms of robustness and discriminating capabilities. Suppose $I(x,y)$ denotes the luminance value of a pixel (x,y) and N_ϕ denote the number of pixels on the line at an angle ϕ. Then, the projected feature vector is given by

$$\mathbf{P} = \begin{bmatrix} P(1^\circ) & P(2^\circ) & \dots & P(180^\circ) \end{bmatrix}^T$$

where

$$P(\phi) = \frac{\sum\limits_{(x,y)\in N_\phi} I^2(x,y)}{N_\phi} - \left(\frac{\sum\limits_{(x,y)\in N_\phi} I(x,y)}{N_\phi} \right)^2, \quad \phi = 1^\circ, 2^\circ, \dots, 180^\circ \tag{2.1}$$

The $\mathbf{P}$ vector is centred around the origin and normalized with respect to the mean μ and standard deviation σ respectively, to obtain the RAV feature vector,

$$\mathbf{R} = \begin{bmatrix} R(1^\circ) & R(2^\circ) & \dots & R(180^\circ) \end{bmatrix}^T$$

where $R(\phi)$ is given by

$$R(\phi) = \frac{(P(\phi) - \mu)}{\sigma}, \quad \phi = 1^\circ, 2^\circ, \dots, 180^\circ \tag{2.2}$$

The 40 low-frequency coefficients of 180-point DCT of the RAV feature vector are chosen to obtain a compact hash vector. Mathematically, the RASH vector

$$\mathbf{D} = \begin{bmatrix} D(1^\circ) & D(2^\circ) & \dots & D(180^\circ) \end{bmatrix}^T$$

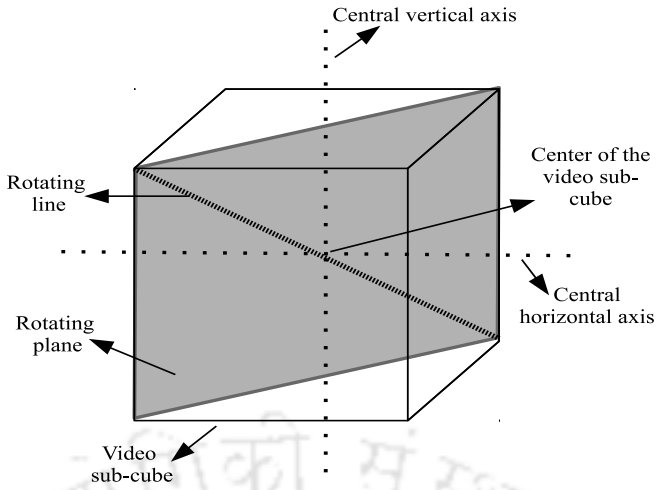


Figure 2.1: Calculation of 3D-RASH vector.

where $D(n)$ is given by,

$$D(n) = \sqrt{\frac{2}{180}} \times \sum_{\phi=1^\circ}^{180^\circ} \left(R(\phi) \times \cos\left(\frac{\pi \times (2\phi + 1) \times n}{2 \times 180} \right) \right), \quad n = 1, 2, \ldots, 40 \tag{2.3}$$

Finally each of the 40 low-frequency DCT coefficients is quantized to 8 bits to obtain a 320-bit image hash.

The extension of the image hashing algorithm to video hashing [10] is as follows. The video sequence is modelled as a collection of video shots, and key frames are selected from those video shots. A shot is identified by selecting the boundary frames. If the ℓ_1 distance between the 64-bin luminance histograms of the current frame and the previous frame is greater than a particular threshold, then the current frame is considered as the boundary frame. Within the frames of a shot, the disparity between the frames is measured [26]. A frame is selected as the keyframe if it minimizes the disparity measurement with its preceding frame. Finally, the hash value is obtained by applying the image hashing algorithm on these keyframes.

The radial projection technique, as proposed in [10], is dependent on keyframes corresponding to a video. This was a major limitation as this technique is exposed to all the non-keyframe-based attacks such as frame dropping, malicious attacks on non-keyframes, and unwanted frame insertions.

2.2 Mathematical Framework of Three Dimensional Radial Projection

Here, we propose a new technique by eliminating the keyframe dependence of the radial projection technique and making the proposed technique dependent on the whole video rather than a particular set of keyframes.

The methods which act in the spatiotemporal domain extract the information from the complete video rather than from selective keyframes. For example, the PARAFAC decomposition based video hashing [15] performs better as far as perceptual hashing for video is concerned.

Here, the information about the video is stored by randomly dividing it into sub-cubes and performing multilinear subspace projection on each of these sub-cubes.

Taking the idea of randomly dividing a video into sub-cubes from the above method, we propose a new *three-dimensional radial projection* (3D-RASH) based perceptual video hash. Here, the video is treated as a cube and the perceptual information is captured using the following procedure. The cube is divided into N randomly located overlapping sub-cubes of size $U \times U \times U$ such that, these sub-cubes cover the entire normalized video of size $X \times X \times X$. Note that $U < X$.

Consider a sub-cube of size $U \times U \times U$ with the centre of the sub-cube lying at the origin $(0, 0, 0)$. We will call this sub-cube as the *reference cube*. Within this reference cube, a line passing through the origin is considered as shown in Figure 2.1. This line is then rotated by 4π steradians (solid angle) in discrete steps so as to cover the entire reference cube. This is achieved by considering a plane containing the origin and passing through the central vertical axis of the reference cube and then rotating the plane by 180° in discrete steps of 1°. Corresponding to each orientation of the plane, a line passing through its centre is rotated by 180° in discrete steps of 1°. In this way, the line rotates by 4π steradians in discrete steps and hence, covers the entire reference cube. Since there are 180 different orientations of plane and corresponding to each orientation, there are 180 different orientations of the line, so in total there are 180×180, i.e. 32400 different orientations of line. Let $L_1, L_2, \ldots, L_{32400}$ represent the lines corresponding to each orientation and n_j be the number of pixels in the jth line where $j = 1, 2, \ldots, 32400$. Corresponding to each line, the coordinates traversed by the line are extracted and retained. Let $\mathbf{c} = \left[(c_{x_1}, c_{y_1}, c_{z_1})\ (c_{x_2}, c_{y_2}, c_{z_2})\ \ldots\ \left(c_{x_{n_j}}, c_{y_{n_j}}, c_{z_{n_j}}\right)\right]^T$ denote the vector comprising of co-ordinates $(c_{x_1}, c_{y_1}, c_{z_1}), (c_{x_2}, c_{y_2}, c_{z_2}), ..., \left(c_{x_{n_j}}, c_{y_{n_j}}, c_{z_{n_j}}\right)$ corresponding to n_j pixels lying on the jth line. Let $l' = \{l_1, l_2, ..., l_{n_j}\}$ denote the array comprising of luminance values $l_1, l_2, ..., l_{n_j}$ corresponding to n_j pixels lying on the jth line. The variance of l' is calculated using

$$\sigma^2 = \frac{1}{n_j} \sum_{i=1}^{n_j} (l_i - \mu)^2 \tag{2.4}$$

where l_i denotes the luminance value of i^{th} pixel lying on the jth line and μ denotes the mean value of all l_is. Thus,

$$\mu = \frac{1}{n_j} \sum_{i=1}^{n_j} l_i \tag{2.5}$$

Corresponding to each of these orientations, there will be a vector comprising of the coordinates lying along the line and a vector comprising of luminance values of the pixels lying on the line. The vectors containing the coordinate values are denoted by $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_{32400}$ and the arrays containing the luminance values of 32400 lines are denoted by $l_1', l_2', ..., l_{32400}'$. Therefore, the array of variance obtained along each orientation of the line from the reference cube using the 3D-radial projection is given by

$$\sigma^2_{(0,0,0)} = \left\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_{180}^2, \sigma_{181}^2, \sigma_{182}^2, \ldots, \sigma_{360}^2, \ldots, \sigma_{32221}^2, \sigma_{32222}^2, \ldots, \sigma_{32400}^2\right\} \tag{2.6}$$

where σ_1^2 denotes the variance of the array $l_1^{'}$, σ_2^2 denotes the variance of the array $l_2^{'}$ and so on.

To generate the array of variance using the 3D-radial projection for an arbitrary sub-cube whose centre coordinates are given by (ι, κ, ζ), the translation operation is applied on the coordinates of the reference cube, using

$$\begin{bmatrix} c_x' \\ c_y' \\ c_z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \iota \\ 0 & 1 & 0 & \kappa \\ 0 & 0 & 1 & \zeta \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_x \\ c_y \\ c_z \\ 1 \end{bmatrix} \tag{2.7}$$

where (c_x, c_y, c_z) represents the co-ordinates of the reference cube and $\left(c_x', c_y', c_z'\right)$ represents the co-ordinates of the translated sub-cube with (ι, κ, ζ) as the centre. After obtaining the transformed coordinates, the luminance values at those corresponding coordinates are extracted and then the above-explained procedure is followed to obtain the array of variance for the translated sub-cube as

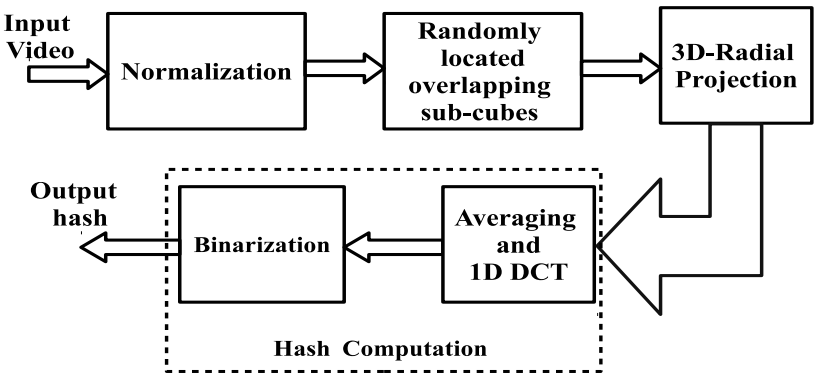
$$\sigma^2_{(\iota,\kappa,\zeta)} = \left\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_{180}^2, \sigma_{181}^2, \sigma_{182}^2, \ldots, \sigma_{360}^2, \ldots, \sigma_{32221}^2, \sigma_{32222}^2, \ldots, \sigma_{32400}^2\right\} \tag{2.8}$$

2.3 Perceptual Video Hashing using Three Dimensional Radial Projection

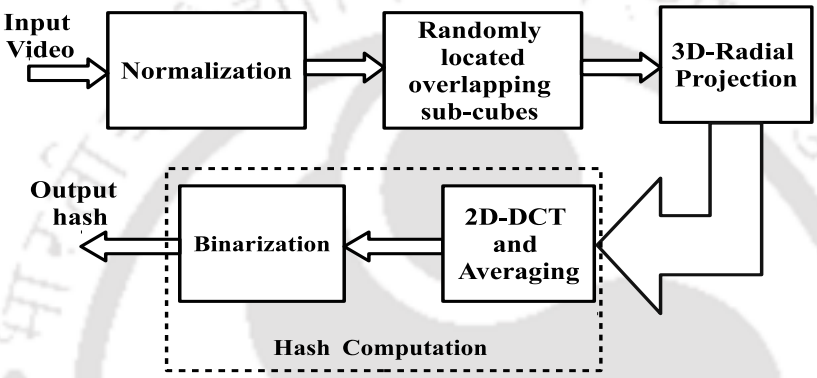
Figure 2.2 shows two schemes for the extraction of the perceptual video hash using the proposed 3D-RASH. In both the schemes the following steps are similar: *normalization*, *selection of randomly located overlapping sub-cubes* and *3D-radial projection*. The remaining steps differ in both the schemes depending on how the perceptual hash can be calculated. The details of these steps are described below:

(i) **Normalization**: Different videos may have different spatial resolutions and a different number of frames. To minimize the effect of these variations on the output hash, it is required to bring the input video, V_{in}, to a standard size in both spatial and temporal direction, say $X \times X \times X$. This is done by down-sampling the pixels of the V_{in} in both spatial and temporal direction. Let V_{norm} be the normalized video considered as a cube.

(ii) **Selection of randomly located overlapping sub-cubes**: N random coordinates within the V_{norm} are generated using a secret key K. From these N random coordinates, the N sub-cubes represented by V_i where $i = 1, 2, \ldots, N$, of size $U \times U \times U$ are formed. The care should be taken that $U < X$ and these N sub-cubes extend over entire V_{norm}. Further, if a different key K' is used then, N different random coordinates are generated and hence the different sub-cubes, leading to a different hash value. This procedure is followed to include security in the process of hash generation.



(a) Scheme-I.



(b) Scheme-II.

Figure 2.2: The block diagram representation for the extraction of perceptual video hash using the three dimensional radial projection.

(iii) **3D-Radial Projection**: The radial projection is applied on the sub-cube as explained in Section 2.2 using the Equation 2.4, the Equation 2.5 and the Equation 2.7 to obtain the array of variance from the sub-cube. The procedure is repeated for all the N randomly located sub-cubes.

(iv) **Hash Computation**: The luminance value of the pixels in a video are highly correlated both in spatial and temporal directions. The variances computed from the neighbouring lines will be highly correlated. This correlation is exploited in Scheme-I and Scheme-II to generate a compact hash. Either of the schemes can be used to generate the perceptual hash from the video.

Scheme-I

Let the following 3D RAV vectors be constructed from the array of variance (Equation

2.8) obtained from the corresponding N sub-cubes of V_{norm} as

$$\begin{aligned} \mathbf{r}_1 &= \left[\sigma_1^2\ \sigma_2^2\ \ldots\ \sigma_{32400}^2\right]^T, \\ \mathbf{r}_2 &= \left[\sigma_1^2\ \sigma_2^2\ \ldots\ \sigma_{32400}^2\right]^T, \\ &\vdots \\ \text{and}\quad \mathbf{r}_N &= \left[\sigma_1^2\ \sigma_2^2\ \ldots\ \sigma_{32400}^2\right]^T \end{aligned}$$

We have now N 3D RAV vectors each of size 32,400. The mean of the vectors give the representative $\mathbf{r}_\mu$ and vectors placed in a matrix $\mathbf{R}_{vec}$, of size $32400 \times N$.

$$\mathbf{R}_{vec} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \ldots & \mathbf{r}_N \end{bmatrix} \tag{2.9}$$

The dimension $32400 \times N$ is very high. One of the easiest way to reduce the dimensionality and at the same time retain the captured features to the best possible extent is by taking the average along the rows of $\mathbf{R}_{vec}$. The average of these RAV vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$ are computed to obtain the final RAV vector $\mathbf{r}_\mu$ using the following equation,

$$\mathbf{r}_\mu = \frac{1}{N} \sum_{i=1}^{N} \mathbf{r}_i \tag{2.10}$$

The final RAV vector $\mathbf{r}_\mu$ is still of high dimension, and this length is not suitable to be considered as the length of the hash. To further reduce the dimensionality the energy compaction property of DCT can be used. Therefore, the 32400 point DCT is applied on $\mathbf{r}_\mu$ to get a transformed vector $\mathbf{t}_{r_\mu}$ as follows

$$\mathbf{t}_{r_\mu} = \mathbf{C}_{DCT}\ \mathbf{r}_\mu \tag{2.11}$$

where $\mathbf{C}_{DCT}$ is the DCT matrix whose elements $\{c\,(i,j)\}$ are given by

$$c(i,j) = \begin{cases} \frac{1}{\sqrt{32400}}, & i = 0; 0 \le j \le 32399 \\ \sqrt{\frac{2}{32400}} \times \cos\left(\frac{\pi(2j+1)i}{2\times 32400}\right), & 1 \le i \le 32399; 0 \le j \le 32399 \end{cases} \tag{2.12}$$

The first few DCT coefficients contain most of the energy, so without introducing much error, we take the first k elements of $\mathbf{t}_{r_\mu}$, where k is the hash length (even). These coefficients are written in the form of a vector which forms the intermediate hash vector $\mathbf{h}'$ corresponding to the given video. To generate the binary hash vector for a given video, obtain the median ξ of $\mathbf{h}'$. The elements of $\mathbf{h}'$ are rank-ordered to obtain $\mathbf{h}'' = [h'_{(1)}\ h'_{(2)} \ldots h'_{(k)}]^T$. The median is calculated as follows

$$\xi = \left(\frac{h'_{\left(\frac{k}{2}\right)} + h'_{\left(\frac{k}{2}\right)+1}}{2} \right) \tag{2.13}$$

Then, elements of $\mathbf{h}'$ are binarized to obtain $\mathbf{h} = [h_1\ h_2 \ldots h_k]^T$ where

$$h_j = \begin{cases} 0; & h'_j < \xi \\ 1; & h'_j \geq \xi \end{cases}, \; j = 1, 2, \ldots, k \tag{2.14}$$

$\mathbf{h}'$ is computed from the feature vector to represent the video. Binarisation by median thresholding results in equal numbers of 1's and 0's in the binarized data. In this case, we can assign a probability of 0.5 for each bit to take the value of 1 and the statistical modelling of binary data becomes easy. The effectiveness of performing the averaging operation in Equation 2.10 and binarizing the elements using the median is discussed in [13].

Scheme-II

The 3D RAV matrices $\mathbf{R}_{mat_i}, \; i = 1, 2, \ldots, N$ are constructed from the array of variance (Equation 2.8) obtained from the corresponding N sub-cubes of V_{norm} as follows

$$\mathbf{R}_{mat_i} = \begin{bmatrix} \sigma^2_1 & \sigma^2_{181} & \cdots & \sigma^2_{32221} \\ \sigma^2_2 & \sigma^2_{182} & \cdots & \sigma^2_{32222} \\ \vdots & \vdots & \cdots & \vdots \\ \sigma^2_{180} & \sigma^2_{360} & \cdots & \sigma^2_{32400} \end{bmatrix}, \quad i = 1, 2, \ldots, N \tag{2.15}$$

Hence, it would be appropriate to use a transformation tool to decorrelate $\mathbf{R}_{mat_i}$ and extract the relevant features to generate a compact hash vector. The two-dimensional (2D) DCT would be a natural choice as it is a simple and effective transformation tool for decorrelating matrix data and capturing most of the energy in low to mid-frequency coefficients. The 2D DCT $\mathbf{T}_i$ of $\mathbf{R}_{mat_i}$ is given by

$$\mathbf{T}_i = 2DDCT\left(\mathbf{R}_{mat_i}\right), \; i = 1, 2, \ldots, N \tag{2.16}$$

The average DCT matrix $\bar{\mathbf{T}}$ is obtained by

$$\bar{\mathbf{T}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{T}_i \tag{2.17}$$

This matrix contains the average information of all the relevant features captured by the 3D RPT and 2D DCT of all the sub-cubes. The low to mid-frequency coefficients sub-matrix of size $k' \times k'$ is selected to get $k = (k')^2$. These k coefficients are vectorized to form the intermediate hash vector $\mathbf{h}'$ for the given video. The hash vector $\mathbf{h}$ is obtained using Equation 2.13 and Equation 2.14.

The hash generation process is described in the Algorithm 2.1.

2.4 Simulation Results and Discussion

A number of experiments are conducted to validate the hash properties and evaluate the performance of the hash. The details of the database, the procedure to validate the hash and the attacks considered for evaluating the performance of the hash discussed below.

Algorithm 2.1 *Proposed video hashing algorithm based on the 3D-RASH.*

1: **Input**:

- Video input V_{in} and k (even).
- Algorithm parameters: U, X, K and N.

2: **Normalization**:

- Normalize V_{in} to $X \times X \times X$ via temporal sub-sampling and spatial re-sizing to obtain $\mathcal{V}_{norm}$.

3: **Selection of randomly located overlapping sub-cubes**:

- Depending on secret key K, randomly select N overlapping 3D sub-cubes from $\mathcal{V}_{norm}$, to form $\mathcal{V}_i$, $i = 1, 2, ..., N$, each of size $U \times U \times U$ such that $U < X$.

4: **3D-Radial Projection**:

- Apply 3D radial projection on entire sub-cube in steps of 1° to calculate the variance of the luminance of the pixels along the projection line. This results in an array of $180 \times 180 = 32,400$ variance values from the sub-cube. The procedure is repeated for all the N randomly located sub-cubes.

5: **Hash Computation**:
Scheme-I

- Construct N 3D RAV vectors from the array of variance obtained from the corresponding N sub-cubes of V_{norm} to obtain $\mathbf{R}_{vec}$ as shown in Equation 2.9.
- Obtain $\mathbf{r}_\mu = \frac{1}{N} \sum_{i=1}^{N} \mathbf{r}_i$
- Reduce the dimensionality using the 1D-DCT on $\mathbf{r}_\mu$ to obtain $\mathbf{t}_{r_\mu} = \mathbf{C}_{DCT}\, \mathbf{r}_\mu$
- Select the first k DCT coefficients to form intermediate hash vector $\mathbf{h}'$.
- Generate the binary hash vector using median based quantization (Equation 2.13 and Equation 2.14).

Scheme-II

- Construct a 3D RAV matrix $\mathbf{R}_{mat_i}$ from the array of variance obtained from the corresponding N sub-cubes of V_{norm} as shown in Equation 2.15.
- Apply DCT to decorrelate and extract the relevant features

$$\mathbf{T}_i = 2DDCT\left(\mathbf{R}_{mat_i}\right),\ i = 1, 2, \ldots, N$$

- Obtain average DCT matrix $\bar{\mathbf{T}}$ from $\mathbf{T}_i$ as shown in Equation 2.17.
- Select low to mid-frequency coefficients sub-matrix of size $k' \times k'$ and vectorize them to form the intermediate hash vector $\mathbf{h}'$ of length $k = (k')^2$ for the given video.
- Binarize $\mathbf{h}'$ using median based quantization (Equation 2.13 and Equation 2.14).

6: **Output**:

- A hash vector $\mathbf{h}$

Table 2.1: Database details used for experimentation.

Parameter	Description
Database	[trecvid.nist.gov], [open-video.org], [reefvid.org], [xiph.org]
Number of videos	1000
Minimum spatial resolution	192×144
Maximum spatial resolution	640×480
Minimum number of frames	66
Maximum number of frames	11,415
Video format	MP4

2.4.1 Database

1000 videos are downloaded from the following video databases: REEFVID [33], XIPH [34], OPEN-VIDEO [35] and TRECVID[36]. The details of the video database are tabulated in Table 2.1.

In both the schemes, videos are normalized to $64 \times 64 \times 64$. 32 randomly overlapping sub-cubes of size $16 \times 16 \times 16$ are chosen depending on a secret key. The number 32, is decided by trial and error to get an optimal performance of the algorithm. The hash length of Scheme-I is 128 while the hash length of Scheme-II is 144. The hash length of all the video hashing algorithms considered for comparison has a hash length around 128.

2.4.2 Attacks

We have classified the attacks into three categories, namely: spatial attacks, temporal attacks and spatiotemporal attacks. The spatial attacks considered are average blurring, Gaussian blurring, the addition of Additive White Gaussian Noise(AWGN), the addition of salt and pepper noise, rotation, cropping (from the borders), brightness modifications, contrast modifications, modifications in the size of spatial resolutions and adding the watermark. The temporal attack considered is regular frame dropping and interpolation. The spatiotemporal attack considered are compression, reverse play and unwanted frame insertion. Further, the combination of spatial attacks and the temporal attacks simultaneously are also considered under the category of the spatiotemporal attack. Figure 2.3 shows a video frame under various attacks considered for the experiments as an example. The brief description of the attacks is as follows:

1. **Blurring**: Spatial averaging and Gaussian filtering were performed on each frame with masks of size 5×5, 9×9, 15×15 and 35×35.

2. **Addition of AWGN**: Zero mean AWGN with standard deviation, σ ranging from 57 to 127, was added to each video frame.

3. **Addition of salt and peeper noise**: the salt and pepper noise was added to each video frame with the following densities: 5%, 10%, 20% and 25%.

4. **Frame rotation**: Each video frame is rotated separately by a small amount in an anti-clockwise direction. The degrees of rotation considered are 2°, 5°, 8° and 12°.

5. **Frame cropping**: 5%, 10% and 15% boundary pixels are cropped from each frame of the video.

6. **Brightness modification**: The brightness of each frame is increased/decreased by adding/-subtracting a value. The values chosen are 5% and 10% of the maximum original frame intensity.

7. **Contrast modification**: The contrast of each frame is increased/decreased by multiplying a value greater/less than one. The values chosen are 0.8, 0.9, 1.10 and 1.20.

8. **Changing the size of spatial resolution**: The spatial resolutions of the original frames were varied to 100×100, 250×250, 500×500 and 1000×1000.

9. **Adding a watermark**: A logo of fixed size was added to each frame of the video. The sizes considered are 16×16, 32×32, 48×48 and 64×64.

10. **Frame dropping and interpolation**: In the first case, only 48 number of the original frames were retained and then the remaining 16 frames were interpolated using temporal averaging. In the second case, only 32 number of the original frames were retained and then the remaining 32 frames were interpolated using temporal averaging.

11. **Compression**: From the original video database in the MP4 format, two more video databases were created with video bit-rates set to 64 kbps and 100 kbps. The average Compression Ratio (CR) of these videos in these two databases are 250:1 and 240:1, respectively. FFMpeg software was used for compressing the videos.

12. **Reverse play**: The frames of the video were played in the reverse order, i.e. from the last frame of the video to the first frame.

13. **Unwanted frame insertion**: After the normalization of the input video four, eight, sixteen and thirty-two frames from a different video (normalized) are inserted to the normalized input video at regular intervals.

2.4.3 Hash Validation

Experiments were conducted to validate the unpredictability, the visual fragility and the perceptual robustness properties of the hash generated using the 3D-RASH based video hashing algorithm. 224 videos from the database and 1000 secret keys are considered to validate the proposed hash. The NHD was the distance metric used to compare the hash values of the videos. Mathematically, the NHD is given by

$$\text{NHD} = \frac{d_H}{k}$$

where d_H and k represent the Hamming distance (HD) between the hashes and length of the hash, respectively. The histograms of the NHD to evaluate the important desirable properties

(a) Original input video frame. (b) Original grey level frame. (c) Rotation by 10°. (d) Compression with average CR 250:1. (e) Gaussian Blurring using a 5×5 mask size.

(f) AWGN with zero mean and $\sigma = 57$. (g) Brightness increased by 10%. (h) Contrast decreased by 10%. (i) 10% Cropping. (j) Watermark attack.

Figure 2.3: Illustrations of a grey level frame under various attacks considered for the experiments.

of the hash generated using Scheme-I and Scheme-II are shown in Figure 2.4 and Figure 2.5 respectively. The validations are described below.

(i) *To validate the unpredictability property*

128-bit hashes were generated for a test sequence using 1000 different secret keys, and then, the NHD was measured between every possible pair of the hashes. The procedure was repeated for the remaining 223 test sequences. The histogram of the NHD values for this experiment is shown in Figure 2.4(a) for Scheme-I and in Figure 2.5(a) for Scheme-II. It can be observed from both the figures, that the NHD measured between the hashes approximately follows a Gaussian distribution with the mean 0.5. The standard deviation of the distribution for Scheme-I is 0.129 and for Scheme-II is 0.118. A narrow-spread distribution (low value of standard deviation) centred around 0.5 implies that the perceptual hash closely satisfies the unpredictability property given in Equation 1.5. Comparing Figure 2.4(a) with Figure 2.5(a) we can conclude that the perceptual hash based on Scheme-II of 3D-RASH satisfies the unpredictability property better than the perceptual hash based on Scheme-I of 3D-RASH.

(ii) *To validate the visual fragility property*

128-bit hashes were generated for 224 test sequences using the same secret key, and then, the NHD was measured between every possible pair of the hashes. The procedure was repeated for the remaining 999 secret keys. The histogram of the NHD values for this experiment is shown in Figure 2.4(b) for Scheme-I and in Figure 2.5(b) for Scheme-II. It can be observed from the figures, that the NHD measured between the hashes approximately follows Gaussian distribution with the mean 0.5. The standard deviation of the distribution for Scheme-I is 0.233 and for Scheme-II is 0.222. A narrow-spread distribution centred

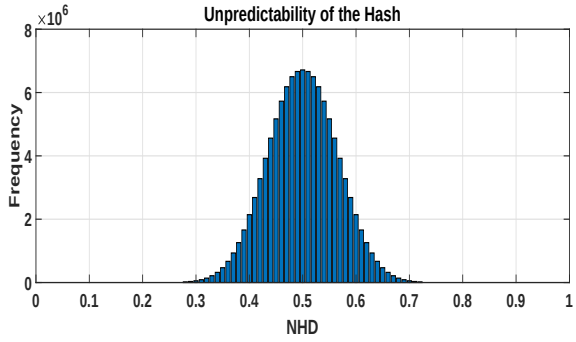
around 0.5 implies that the perceptual hash closely satisfies the visually fragile property given in Equation 1.4. Comparing Figure 2.4(b) and Figure 2.5(b) we can conclude that the perceptual hash based on Scheme-II of 3D-RASH satisfies the visual fragility property better than the perceptual hash based on Scheme-I of 3D-RASH.

(iii) *To validate the perceptual robustness property*
All the videos were corrupted with AWGN of $\sigma = 81$ and subsequently blurred with the Gaussian mask of size 5×5 thereby generating a set of perceptually similar videos. The hashes were generated for these videos using the same secret key. The NHD between the hashes of the original videos and the perceptually similar videos was measured. A histogram concentrated towards zero implies that the perceptual hash closely satisfies the perceptual robustness property given in Equation 1.3. The histogram of all the NHD values for this experiment is shown in Figure 2.4(c) for Scheme-I and in Figure 2.5(c) for Scheme-II. It can be observed from Figure 2.4(c), that the NHD between the original videos and the perceptually similar videos are distributed between 0 and approximately 0.7 with the mean centred around 0.35 for Scheme-I. From Figure 2.5(c) it can be observed that the NHD between the original videos and the perceptually similar videos are are distributed between 0 and approximately 0.65 with the mean centred around 0.325 for Scheme-II. Therefore, we conclude that the perceptual hash based on Scheme-II of 3D-RASH satisfies the perceptual robustness property better than the perceptual hash based on Scheme-I of 3D-RASH.

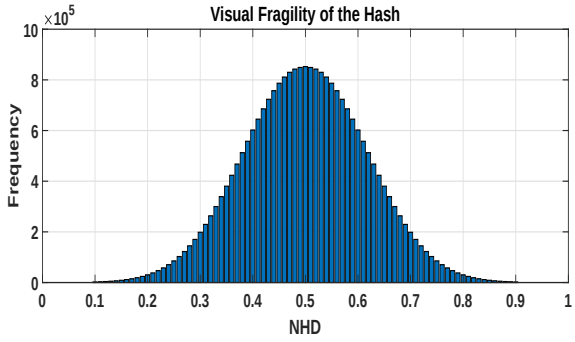
(iv) *To assess perceptual robustness versus visual fragility*
To evaluate the results of perceptual similarity obtained from Scheme-I and Scheme-II of 3D-RASH based video hashing algorithm for the perceptually similar and distinct videos, the histograms of the NHDs obtained for the perceptual robustness property and the visual fragility property are superimposed as shown in Figure 2.4(d) and Figure 2.5(d) respectively. From the figures, it can be observed that the histogram of the NHD for the perceptually similar and distinct videos overlap. The overlapping indicates that there would be a misclassification between the perceptually similar and distinct videos. From Figure 2.4(d) and Figure 2.5(d) it can be observed that there is a large amount of overlapping between the histograms in both the case, however the amount of overlapping is slightly less in Figure 2.5(d) compared to Figure 2.4(d). Hence, it can be concluded that Scheme-II of 3D-RASH will have less misclassification rate compared to Scheme-I of 3D-RASH.

The unpredictability, visual fragility and the perceptual robustness properties of the perceptual hashes generated using Scheme-I and Scheme-II of 3D radial projection are compared in terms of the mean and the standard deviation of the NHD in Table 4.4.

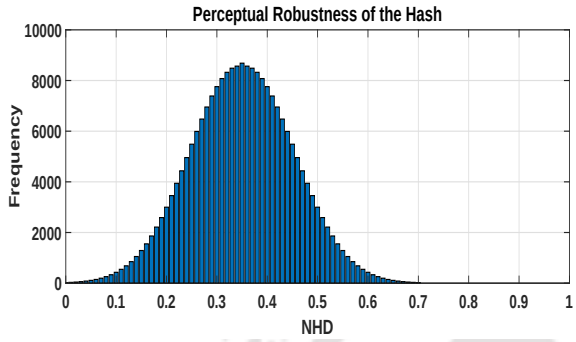
From Figure 2.4, Figure 2.5 and Table 2.2 we conclude that the perceptual hash based on Scheme-I of 3D-RASH is poor than the perceptual hash based on Scheme-II of 3D-RASH. However, the perceptual hash based on Scheme-II of 3D-RASH satisfying the desirable properties of a perceptual hash is found to be average.



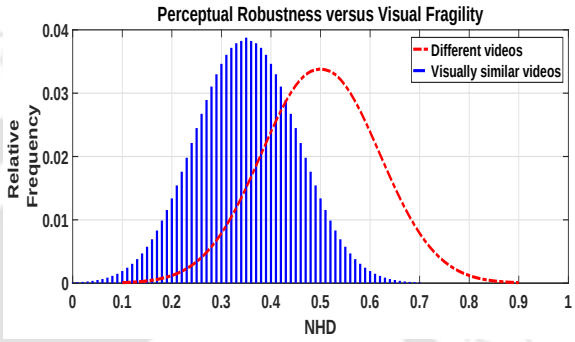
(a) Unpredictability property of the hash.



(b) Visual fragility property of the hash.



(c) Perceptual robustness property of the hash.



(d) Perceptual robustness property versus the visual fragility of the hash.

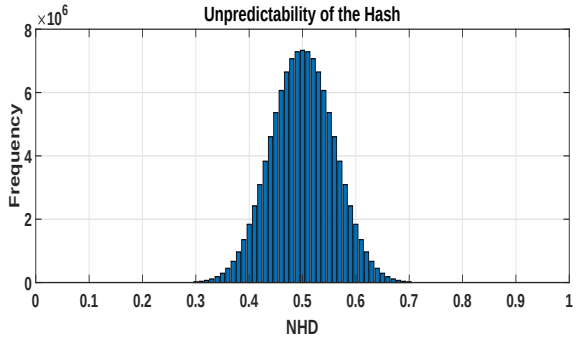
Figure 2.4: The histogram of normalized Hamming distance to evaluate the important desirable properties of the hash generated using Scheme-I of 3D-RASH based video hashing.

Table 2.2: Comparison of hash validity among the two schemes of 3D-RASH based perceptual video hashing algorithm.

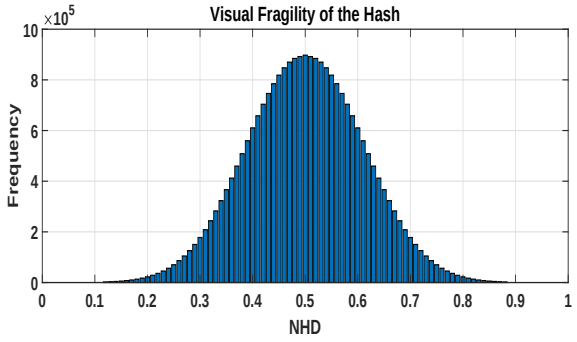
Name of the Video Hashing Algorithm	Unpredictability		Visual Fragility		Perceptual Robustness	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
3D-RASH - Scheme-I	**0.5**	**0.129**	**0.5**	**0.233**	**0.35**	**0.204**
3D-RASH - Scheme-II	**0.5**	**0.118**	**0.5**	**0.222**	**0.325**	**0.190**

2.4.4 Evaluation of Hash Performance

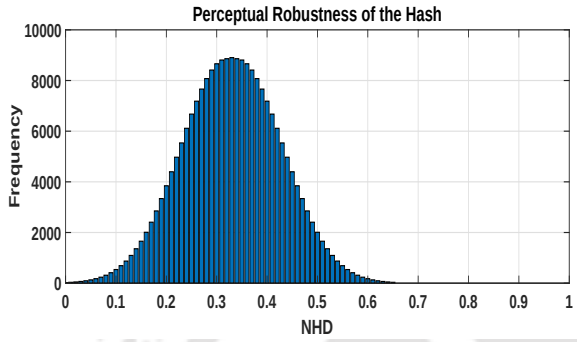
The receiver operating characteristic (ROC) curves [37] were used to evaluate the performance of the algorithm. An ROC curve is a plot of *miss probabilities* at different *false-alarm probabilities*. If V represents the reference video, V_{sim} represents a video perceptually similar to V and V_{diff} represents a video perceptually different from V then, the miss probability is defined as the probability of V_{sim} not being classified as V and the false-alarm probability is defined as the probability of V_{diff} classified as V. The best possible prediction would yield the point (0, 0) as it refers to miss probability = 0 and false-alarm probability = 0. The diagonal line passing through (0, 1) and (1, 0) is called the line of no discrimination as any point which lies on this line is as good as the random prediction. The ROC space can be split into two regions based on this diagonal line. The region below the diagonal implies that the classification result is good, in other words, the classification is better than random. The region above the diagonal implies that the classification result is poor. In other words, the classification is worse than random



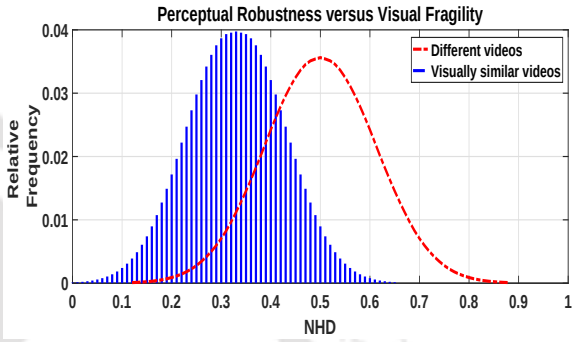
(a) Unpredictability property of the hash.



(b) Visual fragility property of the hash.



(c) Perceptual robustness property of the hash.



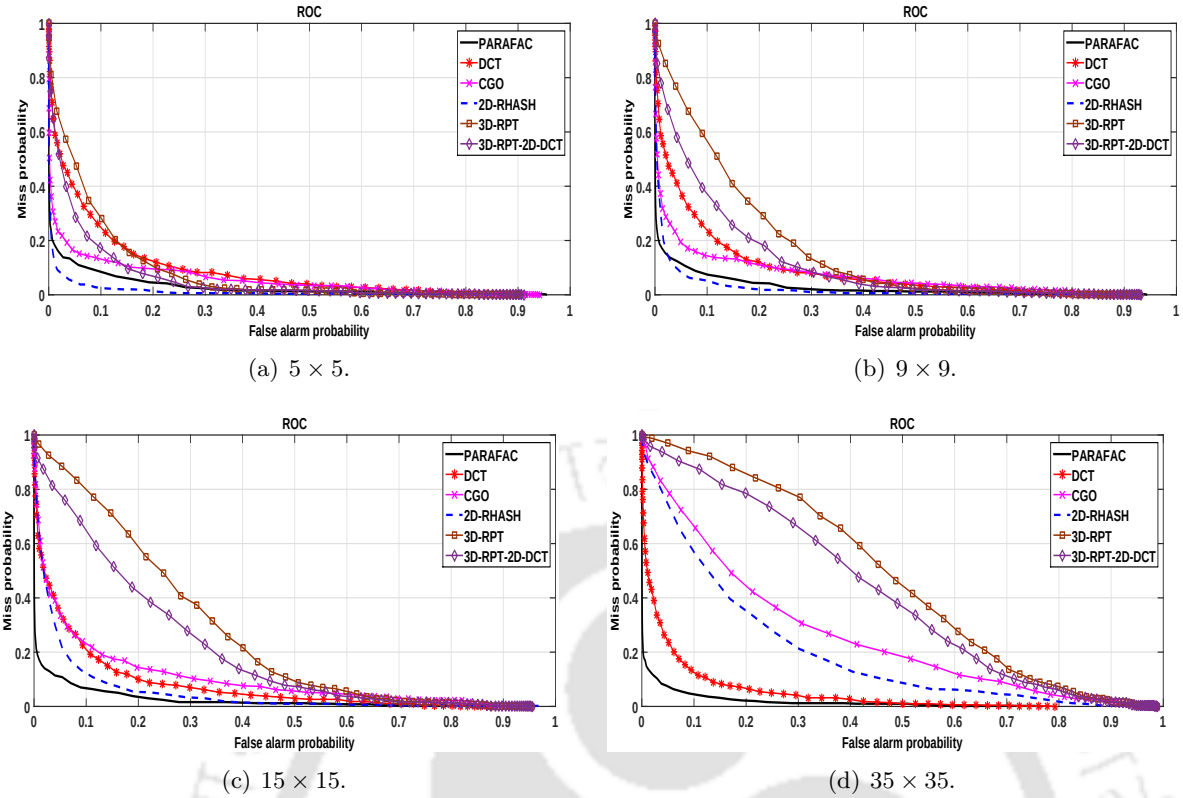
(d) Perceptual robustness property versus the visual fragility of the hash.

Figure 2.5: The histogram of normalized Hamming distance to evaluate the important desirable properties of the hash generated using Scheme-II of 3D-RASH based video hashing.

prediction. We will call the region below the diagonal as *good region* and the region above the diagonal as *bad region*. If we plot the ROC curves corresponding to different perceptual video hashing techniques on the same graph then the lowest curve which is nearest to the point (0, 0) will correspond to the best perceptual video hashing technique. Similarly, the highest curve which is farthest from the point (0, 0) will correspond to the poor video hashing technique.

The following perceptual video hashing algorithms were considered for comparison:

- the PARAFAC decomposition based video hashing algorithm [15], labelled as PARAFAC in the ROC curve.
- the DCT based video hashing algorithm [12, 13], labelled as DCT in the ROC curve.
- the CGO based video hashing algorithm [7, 27], labelled as CGO in the ROC curve.
- the 2D radial projection based video hashing algorithm [11, 10], labelled as 2D-RHASH in the ROC curve.
- the video hashing algorithm based on the 3D-RASH using Scheme-I and Scheme-II are respectively labelled as 3D-RPT and 3D-RPT-2D-DCT in the ROC curve.



(a) 5×5. (b) 9×9.

(c) 15×15. (d) 35×35.

Figure 2.6: Statistical evaluation of video hashing algorithms via the ROC curves under average blurring attack using different mask size.

2.4.4.1 Single Attacks

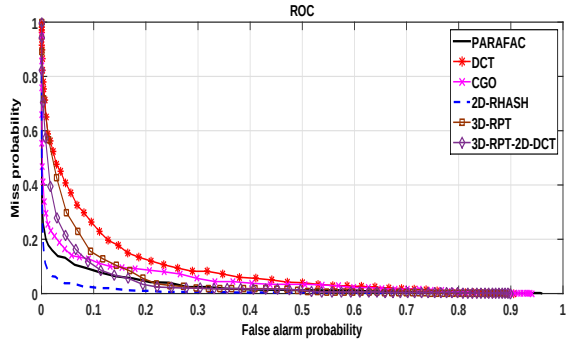
The evaluation of hash performance for various single attacks is as follows. The attacks include the image processing attacks and the malicious attacks described earlier.

Average blurring attack: Figure 2.6 shows the ROC curves under average blurring using the mask of the following sizes: 5×5, 9×9, 15×15 and 35×35. The performance of the video hashing algorithms based on the PARAFAC decomposition and the 3D-DCT is relatively stable. The performance of the video hashing algorithm based on the 2D-RASH, the CGO, Scheme-I and Scheme-II of the 3D-RASH deteriorate for 15×15 and 35×35 mask sizes.

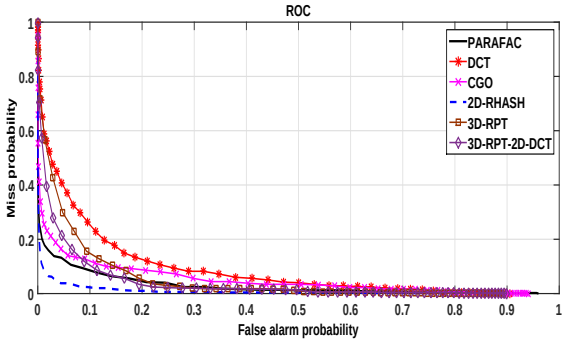
Gaussian blurring attack: The ROC curves under the Gaussian filtering using the 5×5, 9×9, 15×15 and 35×35 masks are shown in Figure 2.7. The performance of all the video hashing algorithms under considerations is similarly good for all the mask sizes. It can be observed that the performance of all the video hashing algorithms considered deteriorates as the mask size is increased to 15×15 and 35×35.

AWGN attack: Figure 2.8 shows the ROC curves for the zero mean AWGN attack with the following values of σ: 57, 81, 114 and 127. The performance of the video hashing algorithms based on the PARAFAC decomposition and the 3D-DCT is relatively stable. As the value of σ increases, the performance of the video hashing algorithms based on the 2D-RASH, Scheme-I and Scheme-II of the 3D-RASH deteriorates quickly.

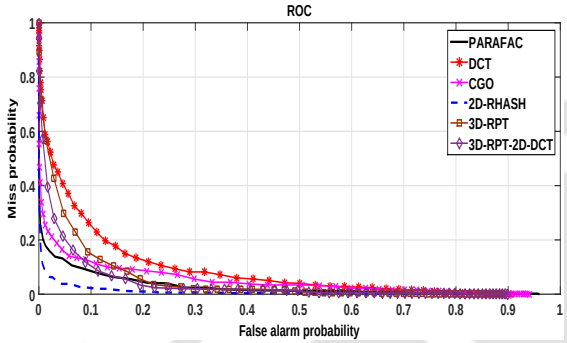
Salt and pepper noise attack: Figure 2.9 shows the ROC curves under the salt and pepper noise attack with the following noise densities: 5%, 10%, 20% and 25%. The perfor-



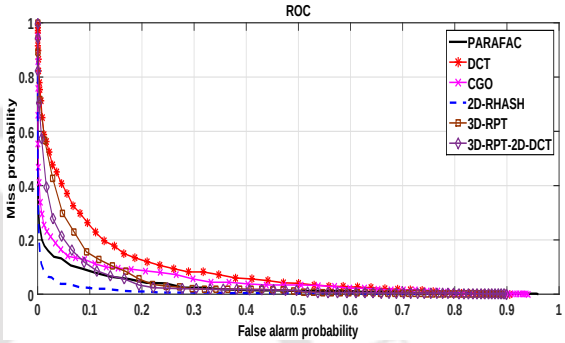
(a) 5×5.



(b) 9×9.

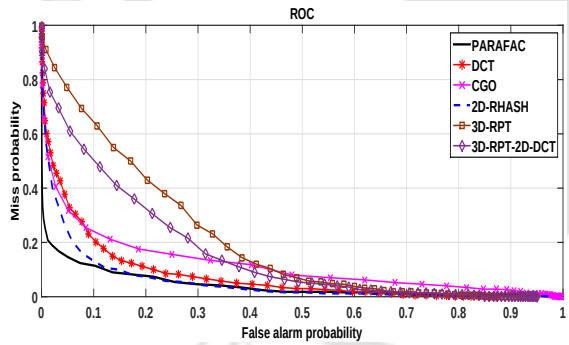


(c) 15×15.

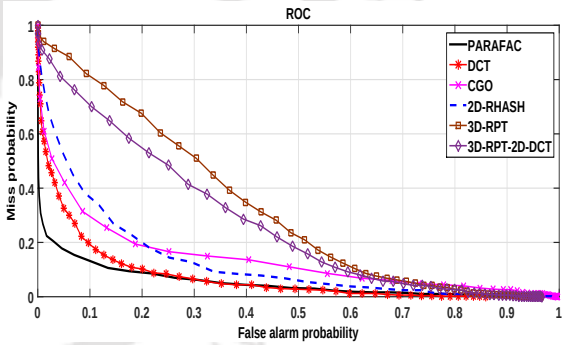


(d) 35×35.

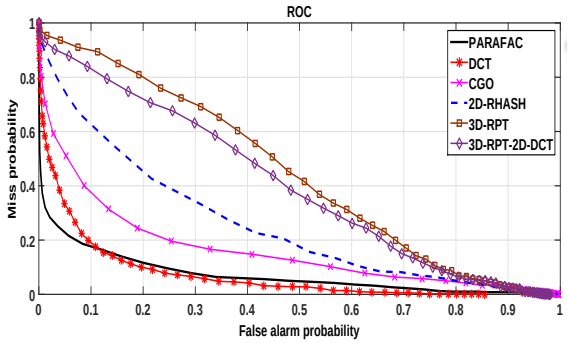
Figure 2.7: Statistical evaluation of video hashing algorithms via the ROC curves under Gaussian blurring attack using different mask size.



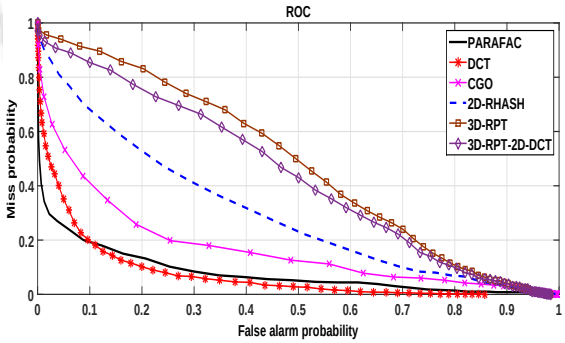
(a) Zero mean AWGN with $\sigma = 57$.



(b) Zero mean AWGN with $\sigma = 81$.

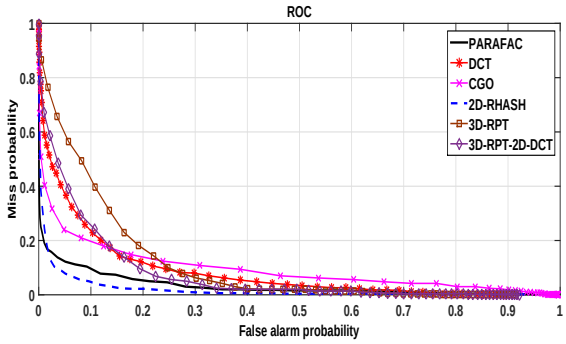


(c) Zero mean AWGN with $\sigma = 114$.

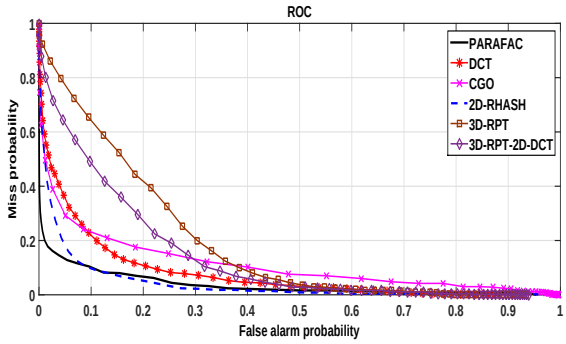


(d) Zero mean AWGN with $\sigma = 127$.

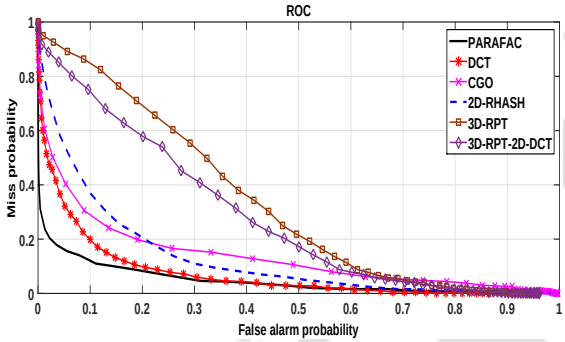
Figure 2.8: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN attack.



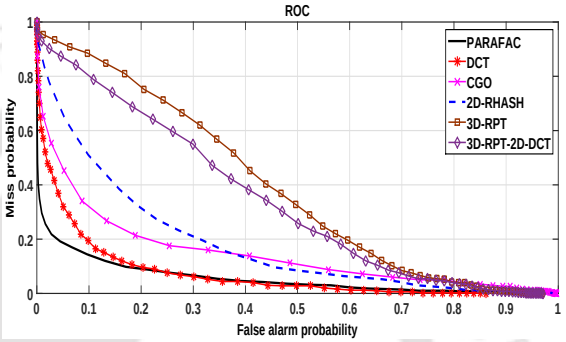
(a) Salt and pepper noise with 5% noise density.



(b) Salt and pepper noise with 10% noise density.



(c) Salt and pepper noise with 20% noise density.



(d) Salt and pepper noise with 25% noise density.

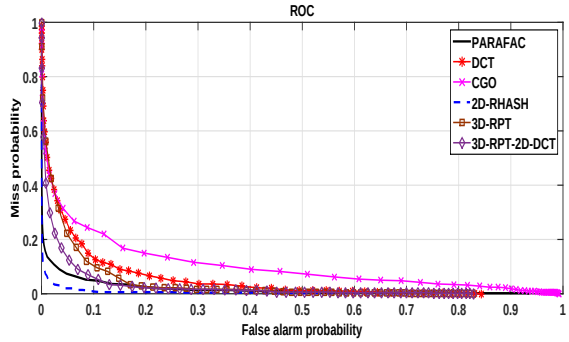
Figure 2.9: Statistical evaluation of video hashing algorithms via the ROC curves under salt and pepper noise attack.

mance of the video hashing algorithms based on the PARAFAC decomposition and 3D-DCT is relatively stable. It can be observed that as the density of the noise increases, the performance of the video hashing algorithm based on the 2D-RASH, Scheme-I and Scheme-II of 3D-RASH degrades quickly.

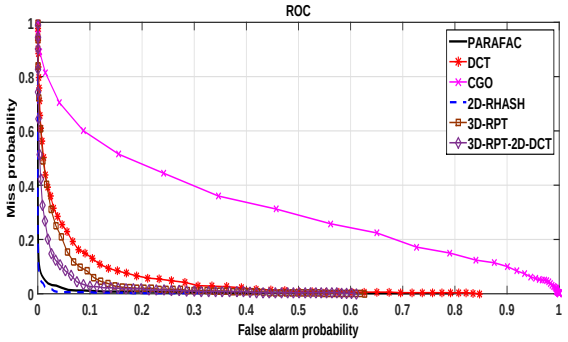
Rotation attack: Figure 2.10 shows the ROC curves under the rotation attack with the following degrees of rotation: 2°, 5°, 8° and 12°. The performance of all the video hashing algorithms considered is nearly same for 2° and 5° rotation attack. For 8° and 12° rotation attack, the 3D-DCT based video hashing algorithm shows poor performance while the CGO based video hashing algorithm shows the least robustness to the rotation attack beyond 2°. The performance of the video hashing algorithm based on Scheme-I and Scheme-II of 3D-RASH is relatively stable for all the degree of rotation attack considered.

Cropping attack: The ROC curves for the cropping attack are shown in Figure 2.11. The following percentages of pixels were cropped from the borders of each frame: 5%, 10% and 15%. The performance of all the video hashing algorithms considered is good and relatively same except the 3D-DCT based video hashing algorithm. It can be observed from the figure that as more and more boundary pixels are cropped, the performance of the video hashing algorithm based on the 3D-DCT becomes progressively poor compared to the other video hashing algorithms. It can be observed that the 2D-RASH based video hashing algorithm followed by the other video hashing algorithms considered are relatively stable for the cropping attack.

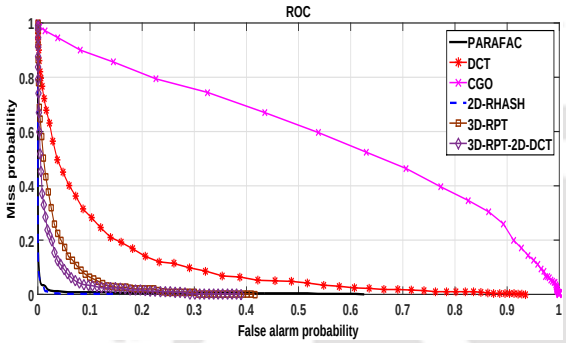
Brightness variation attack: The ROC curves under the modified brightness attack



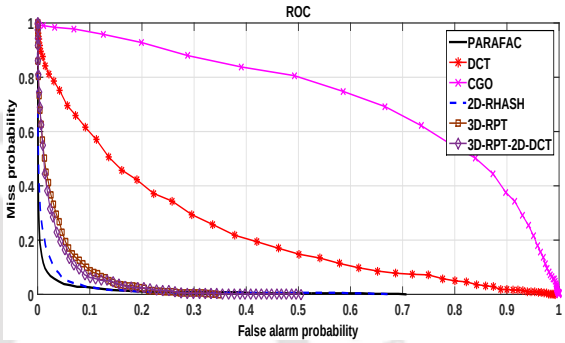
(a) 2°.



(b) 5°.

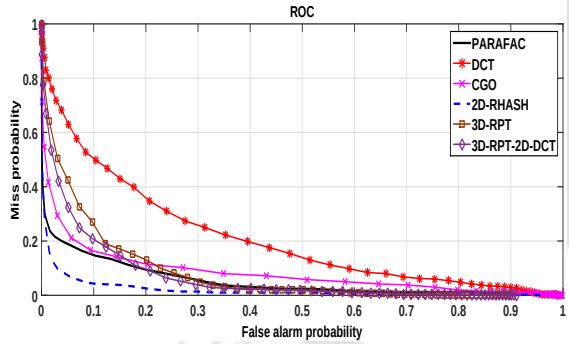


(c) 8°.

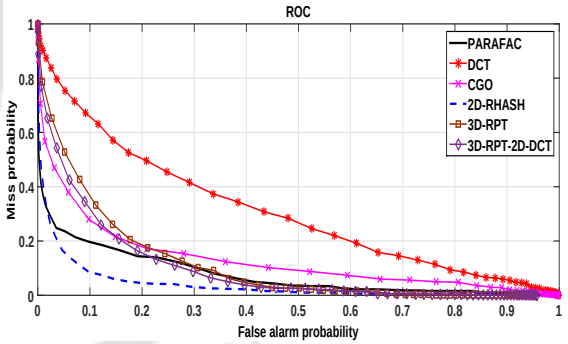


(d) 12°.

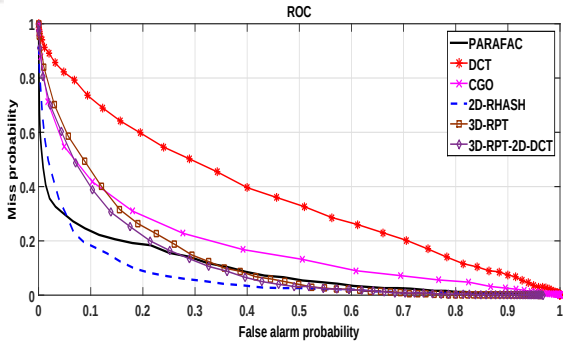
Figure 2.10: Statistical evaluation of video hashing algorithms via the ROC curves under rotation attack with different degrees of frame rotation.



(a) 5% crop.

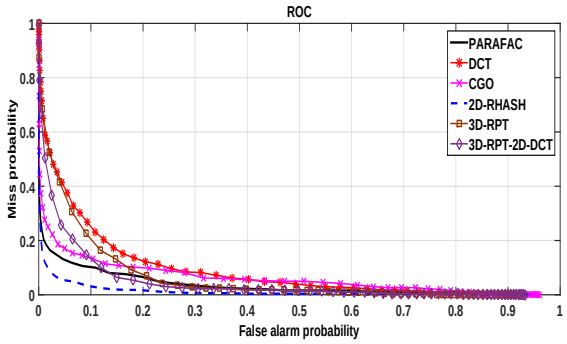


(b) 10% crop.

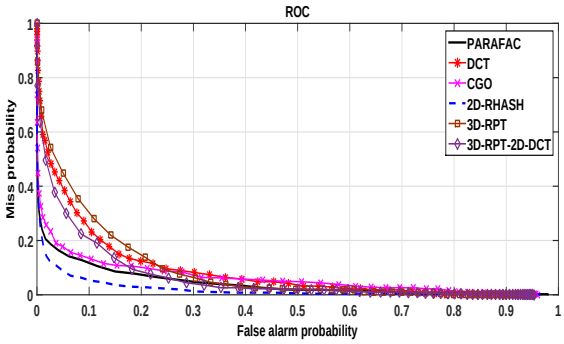


(c) 15% crop.

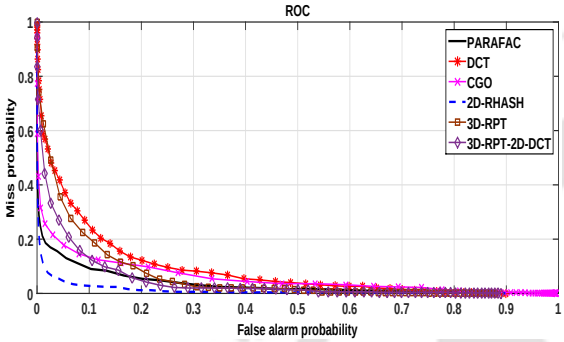
Figure 2.11: Statistical evaluation of video hashing algorithms via the ROC curves under cropping attack with different percentages of crop.



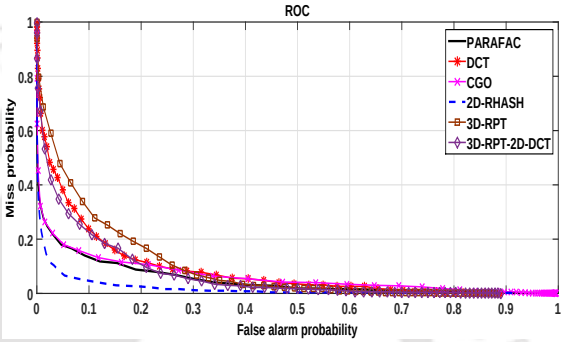
(a) 5% brightness increase.



(b) 10% brightness increase.



(c) 5% brightness decrease.



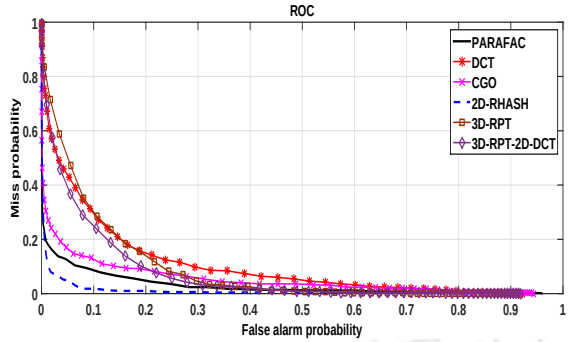
(d) 10% brightness decrease.

Figure 2.12: Statistical evaluation of video hashing algorithms via the ROC curves under brightness variation attack.

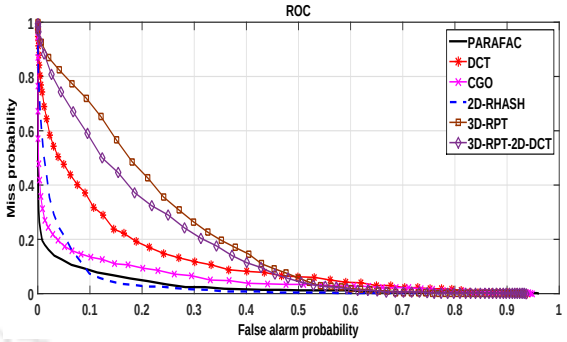
($\pm$5% and $\pm$10%) are shown in Figure 2.12. The performance of the 2D-RASH based video hashing algorithm is relatively stable. The performance of all the other video hashing algorithms considered is similar and deteriorates for $\pm$10% modification in the brightness.

Contrast variation attack: The ROC curves for the modifications in the contrast are shown in Figure 2.13. The contrast of the frames is increased by multiplying the frame pixels with 1.1 and 1.2, and decreased by multiplying the frame pixels with 0.9 and 0.8, respectively. The performance of the PARAFAC decomposition based video hashing algorithm is relatively stable for the modified contrast attack followed by the video hashing algorithms based on the CGO and the 3D-DCT. The performance of the other video hashing algorithms considered is robust to the increased contrast attack while susceptible to the decreased contrast attack. The video hashing algorithms based on the 2D-RASH, Scheme-I and Scheme-II of 3D-RASH are least robust to the contrast decrease attack.

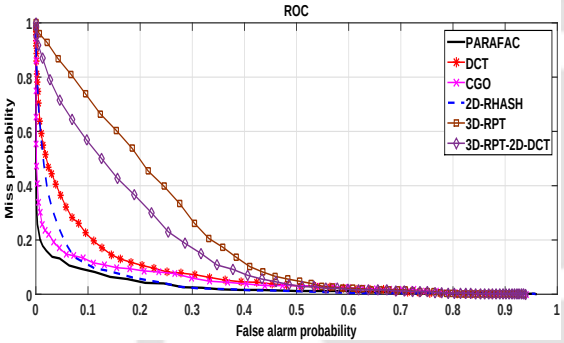
Frame-drop and interpolation attack: The ROC curves for the frame-drop attack are shown in Figure 2.14. In the first case, only 48 frames were retained, and then the remaining 16 frames were interpolated using temporal averaging. It can be observed that the video hashing algorithms based on the 2D-RASH, the CGO and the PARAFAC decomposition show similar performance and has the least area under the ROC curves followed by the video hashing algorithm based on the Scheme-I and Scheme-II of 3D-RASH and the 2D-DCT. In the second case, only 32 frames were retained, and then the remaining 32 frames were interpolated using temporal averaging. In this case, all the video hashing algorithms considered show similar



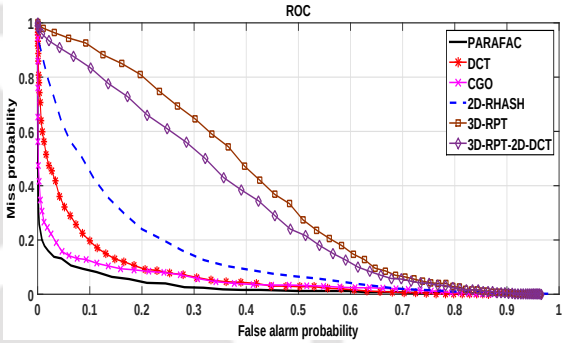
(a) 10% contrast increase.



(b) 20% contrast increase.

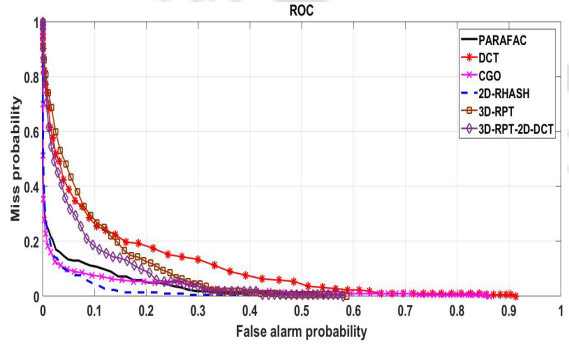


(c) 10% contrast decrease.

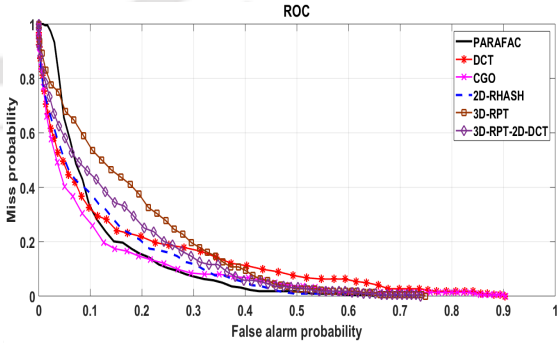


(d) 20% contrast decrease.

Figure 2.13: Statistical evaluation of video hashing algorithms via the ROC curves under contrast variation attack.

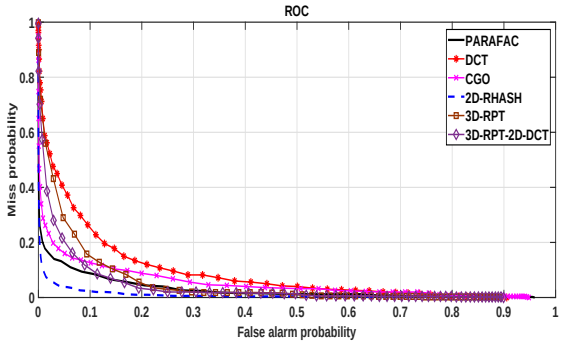


(a) 48 frames retained.

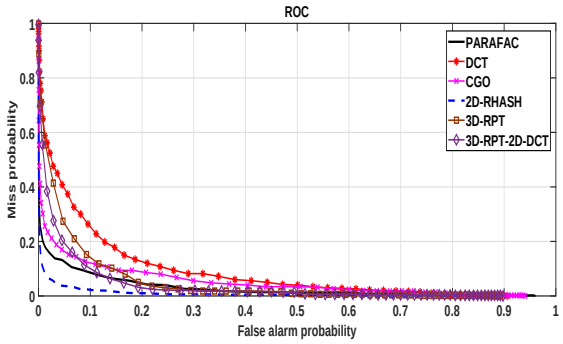


(b) 32 frames retained.

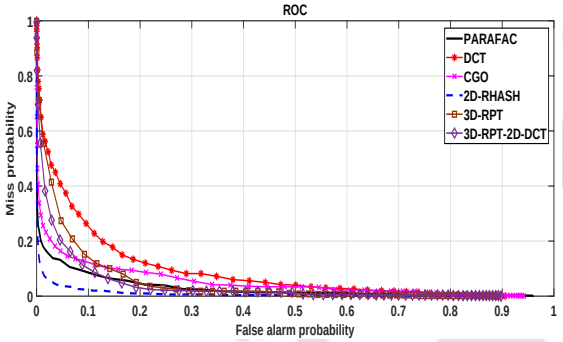
Figure 2.14: Statistical evaluation of video hashing algorithms via the ROC curves under frame-drop and interpolation attack.



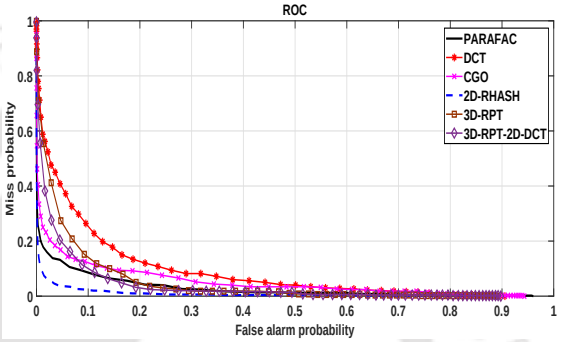
(a) 100×100.



(b) 250×250.

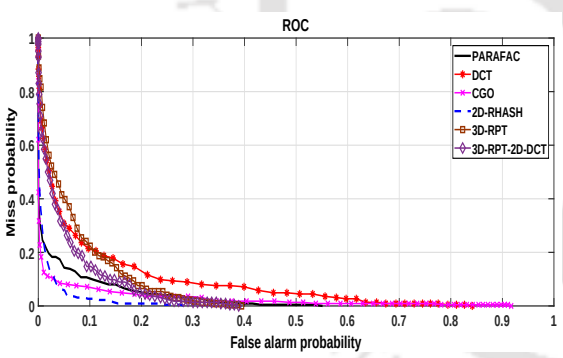


(c) 500×500.

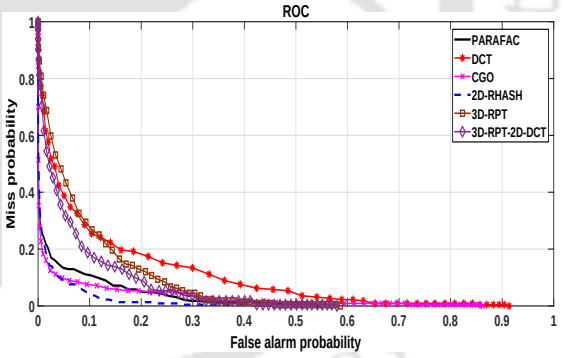


(d) 1000×1000.

Figure 2.15: Statistical evaluation of video hashing algorithms via the ROC curves under spatial resolution variation attack.



(a) 100kbps bit-rate and 240:1 average CR.



(b) 64kbps bit-rate and 250:1 average CR.

Figure 2.16: Statistical evaluation of video hashing algorithms via the ROC curves under compression attack.

performance.

Spatial resolution variation attack: Figure 2.15 shows the ROC curves under the modified spatial resolutions attack. The following spatial resolutions were considered: 100×100, 250×250, 500×500 and 1000×1000. It can be observed that all the algorithms under considerations show similar performance. The 2D-RASH based video hashing algorithm is more robust to the variations in the spatial resolution while the 3D-DCT based video hashing algorithm is less robust to this attack.

Compression attack: Two video database was constructed with the average CR of 250:1

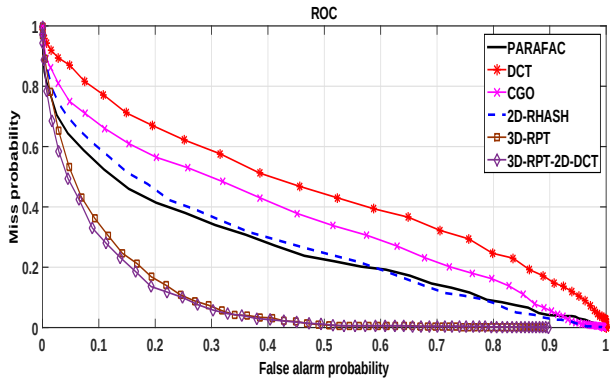
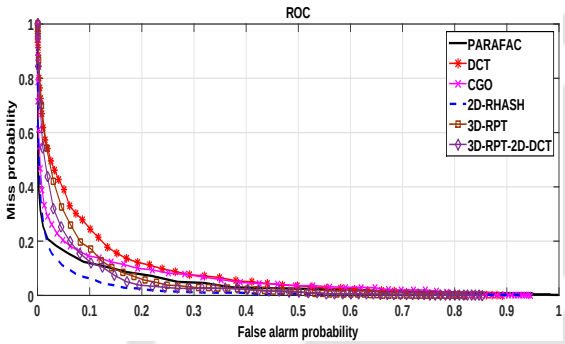
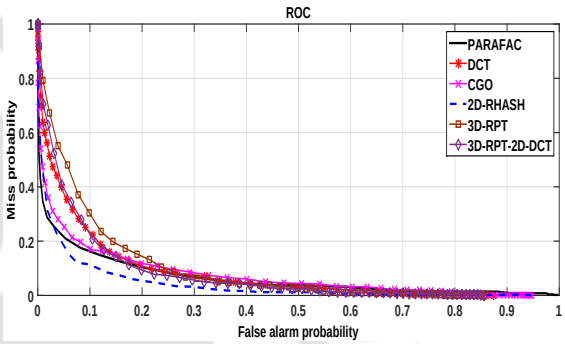


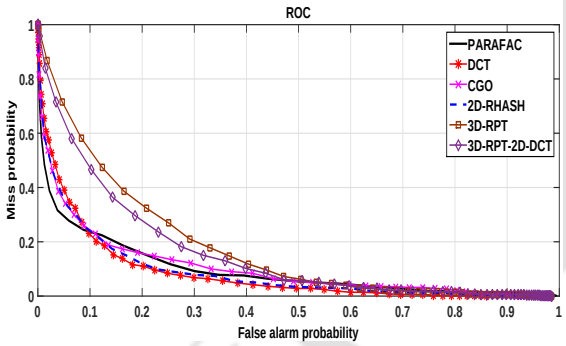
Figure 2.17: Statistical evaluation of video hashing algorithms via the ROC curves under reverse play attack.



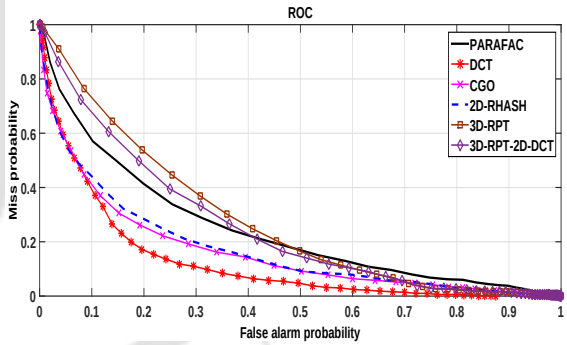
(a) 4 unwanted frames inserted.



(b) 8 unwanted frames inserted.



(c) 16 unwanted frames inserted.

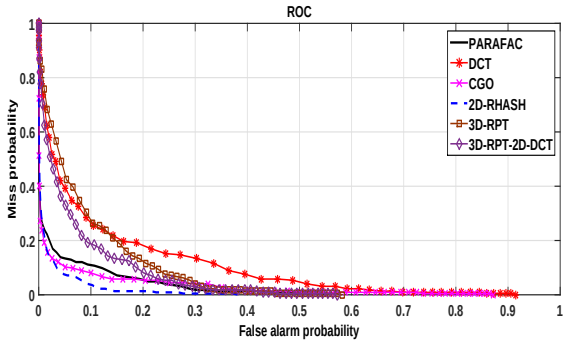


(d) 32 unwanted frames inserted.

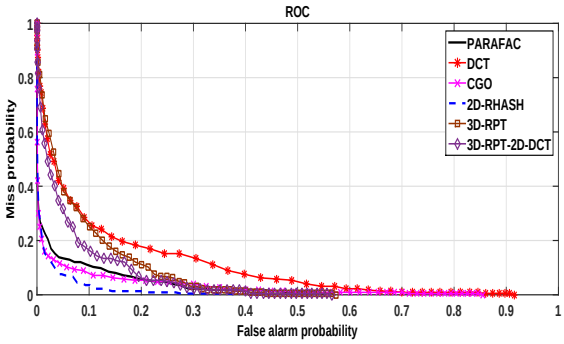
Figure 2.18: Statistical evaluation of video hashing algorithms via the ROC curves under frame insertion attack.

and 240:1, respectively. The statistical evaluation of video hashing algorithms using the ROC curves under the compression attack is shown in Figure 2.16. It can be observed that all the algorithms under consideration show similar performance.

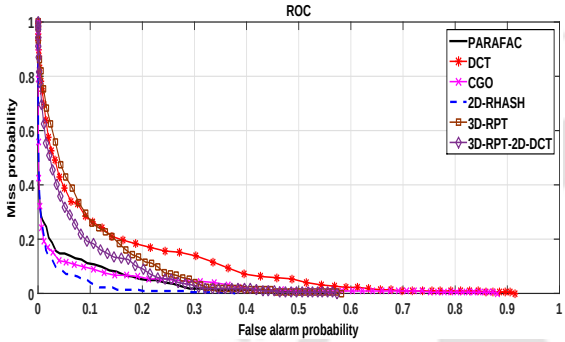
Reverse play attack: The ROC curves under the reverse play attack are shown in Figure 2.17. The video hashing algorithm based on Scheme-I and Scheme-II of 3D-RASH is robust to this attack because of the symmetry property of the variance i.e. the variance of $\{a, b, c\}$ is equal to variance of $\{c, b, a\}$. The ROC curves for the remaining video hashing algorithms stayed close to the line of no discrimination implying the poor performance of all the algorithms considered.



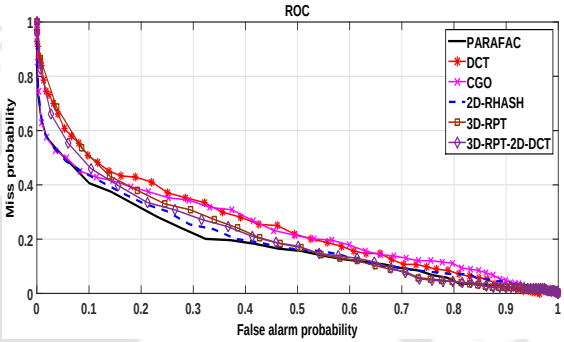
(a) Watermark of size 16×16.



(b) Watermark of size 32×32.



(c) Watermark of size 48×48.

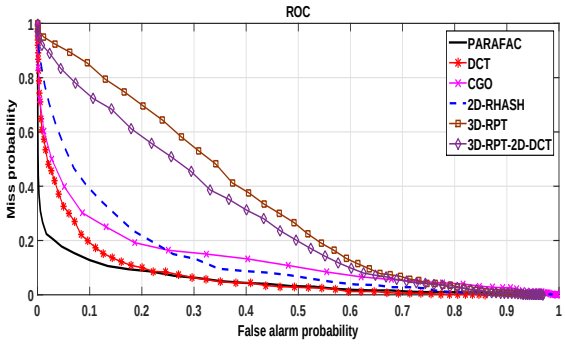


(d) Watermark of size 64×64.

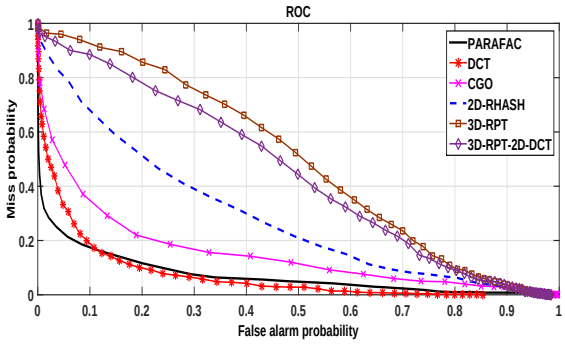
Figure 2.19: Statistical evaluation of video hashing algorithms via the ROC curves under watermark insertion attack.

Frame insertion attack: Firstly, four frames of a different video are inserted in the normalized version of the original video, and the performance of the algorithms is evaluated. This attack is repeated by inserting eight, sixteen and thirty-two unwanted frames, and then the performance of the algorithms is evaluated. The ROC curves under this attack are shown in Figure 2.18. This is a malicious attack, and hence the video hashing algorithm having the largest area under the ROC curve is considered to be performing better. When 4 or 8 different video frames are inserted, all the video hashing algorithms has failed to capture the malicious attack because of being perceptually robust. As the number of unwanted frame insertion increases, the video hashing algorithm based on Scheme-I and Scheme-II of 3D-RASH perform better followed by the video hashing algorithms based on the PARAFAC decomposition, the 2D-RASH, the CGO and the 3D-DCT.

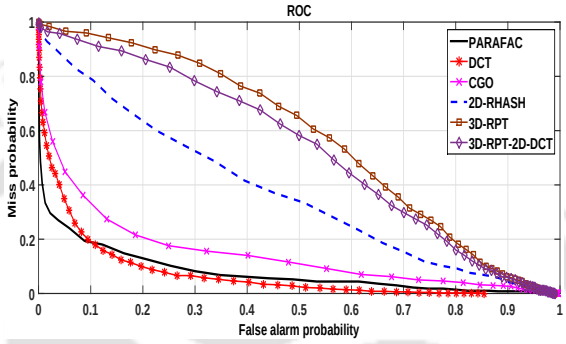
Watermark insertion attack: In this attack, a visible logo of different sizes is inserted to each video frame as a visible watermark. The following sizes of the logo are considered: 16×16, 32×32, 48×48 and 64×64. The ROC curves for this attack are shown in Figure 2.19. This is also considered to be a malicious attack, and hence the video hashing algorithm having the largest area under the ROC curve is considered to be performing better. For the size of the logo up to 48×48 the performance of the video hashing algorithms based on the 2D-RASH, the CGO and the PARAFAC decomposition is having less area under the ROC curve while the video hashing algorithms based on Scheme-I and Scheme-II of 3D-RASH and the 3D-DCT is have more area under the ROC curve. For the logo of size 64×64, the performance of all the



(a) Average blurring using a 3×3 mask and adding zero mean AWGN with $\sigma = 81$.



(b) Average blurring using a 5×5 mask and adding zero mean AWGN with $\sigma = 114$.



(c) Average blurring using a 9×9 mask and adding zero mean AWGN with $\sigma = 127$.

Figure 2.20: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN and average blurring attacks.

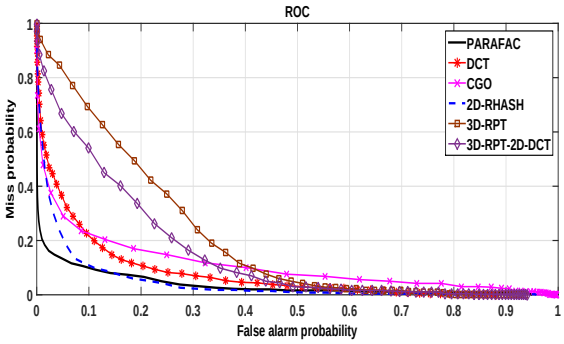
video hashing algorithms is similarly good.

2.4.4.2 Multiple Attacks

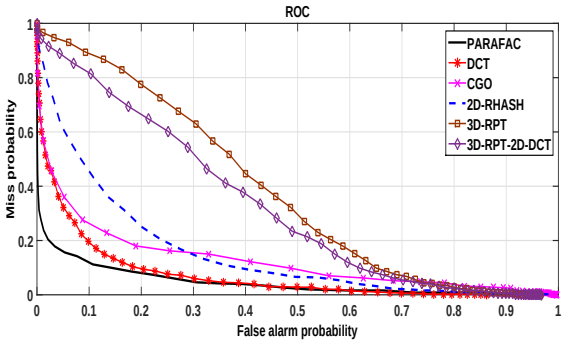
The evaluation of hash performance for various multiple attacks is as follows. The attacks include combination of various single image processing attacks and malicious attacks described earlier.

AWGN and average blurring attacks: In this attack, each video frame was subjected to the average blurring with different mask sizes (3×3, 5×5 and 9×9), and then the addition of the zero mean AWGN with σ ranging from 81 to 127. The corresponding ROC curves are shown in Figure 2.20. The performance of the video hashing algorithms based on the 2D-RASH, Scheme-I and Scheme-II of the 3D-RASH degrades quickly as the value of σ increases. The performance of the video hashing algorithms based on the PARAFAC decomposition, the 3D-DCT and the CGO is relatively stable.

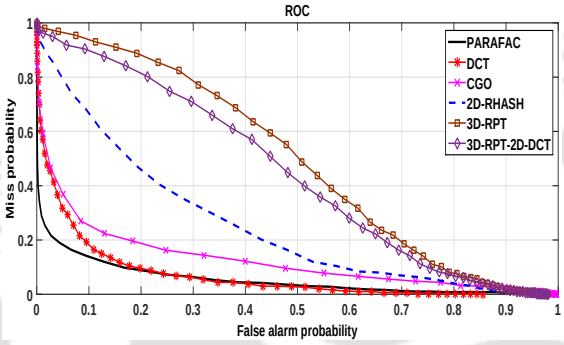
Salt and pepper noise and average blurring attacks: In this attack, each video frame was subjected to the average blurring with different mask sizes (3×3, 5×5 and 9×9), and then the addition of the salt and pepper noise with densities ranging from 5% to 25%. The corresponding ROC curves are shown in Figure 2.21. The performance of the video hashing algorithms based on the 2D-RASH, Scheme-I and Scheme-II of 3D-RASH degrades quickly as the percentage of noise density increases. The performance of the video hashing algorithms



(a) Average blurring using a 3×3 mask and adding a salt and pepper noise with 5% noise density.



(b) Average blurring using a 5×5 mask and adding a salt and pepper noise with 20% noise density.



(c) Average blurring using a 9×9 mask and adding a salt and pepper noise with 25% noise density.

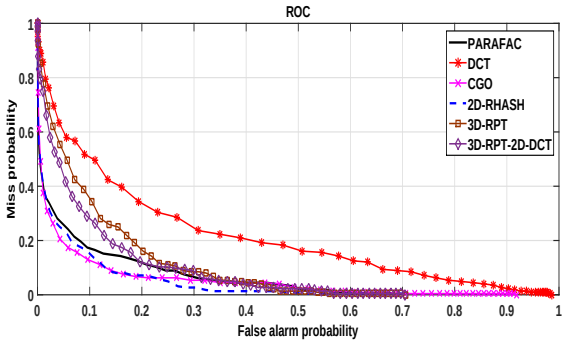
Figure 2.21: Statistical evaluation of video hashing algorithms via the ROC curves under salt and pepper noise and average blurring attacks.

based on the PARAFAC decomposition, the 3D-DCT and the CGO is relatively stable.

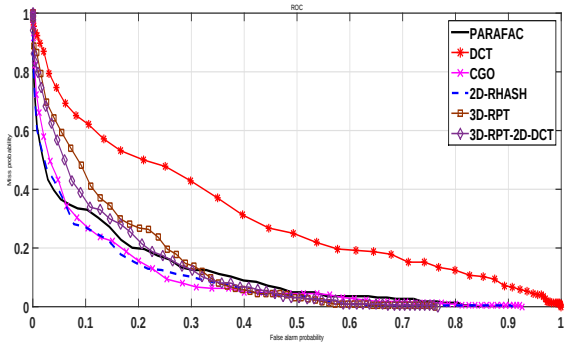
Cropping and Gaussian blurring attacks: In this attack, each video frame was subjected to the Gaussian filtering with different mask sizes (3×3, 5×5 and 9×9), and then the cropping of frame boundary pixels (5%, 10% and 15%). The corresponding ROC curves are shown in Figure 2.22. In all the cases, the performance of all the video hashing algorithms is good except for the 3D-DCT based video hashing algorithm showing relatively poor performance.

AWGN and Gaussian blurring attacks: In this attack, each video frame was corrupted with zero mean AWGN with σ ranging from 57 to 114 and followed by the Gaussian filtering with different mask sizes (3×3, 5×5 and 9×9). The corresponding ROC curves are shown in Figure 2.23. It can be observed that in all the three cases, the performance of the video hashing algorithms based on the 2D-RASH, the PARAFAC decomposition and the 3D-DCT is relatively stable. Further, the performance of the video hashing algorithms based on the CGO, Scheme-I and Scheme-II of the 3D-RASH is relatively unstable and poor.

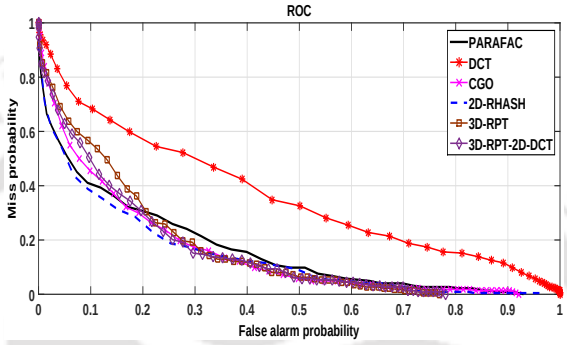
Frame-dropping and interpolation followed by rotation attacks: In this attack video frames are dropped at regular intervals while retaining only 16 or 8 of the original frames and then interpolated to obtain 64 number of frames, followed by 5° or 8° rotation attacks respectively. The ROC curves for these multiple attacks are shown in Figure 2.24. In the first case of frame dropping and interpolation and the rotation attacks, the performance of all the



(a) Gaussian blurring using a 3×3 mask and 5% cropping.



(b) Gaussian blurring using a 5×5 mask and 10% cropping.



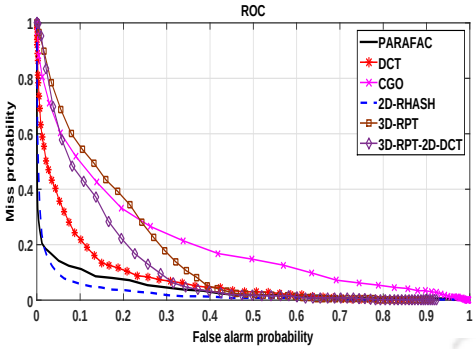
(c) Gaussian blurring using a 9×9 mask and 15% cropping.

Figure 2.22: Statistical evaluation of video hashing algorithms via the ROC curves under cropping and Gaussian blurring attacks.

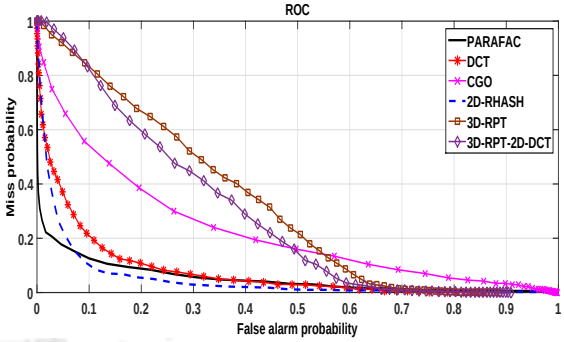
video hashing algorithms is good except for the CGO based video hashing algorithm. In the second case of frame dropping and interpolation and the rotation attacks, the video hashing algorithm based on Scheme-I and Scheme-II of the 3D-RASH is relatively stable followed by the 2D-RASH based video hashing algorithm. The performance of the video hashing algorithms based on the PARAFAC decomposition, the 3D-DCT and the CGO is relatively poor.

Increased spatial resolution, frame-dropping and interpolation followed by rotation attacks: In this attack, the spatial resolution of each frame was increased to 500×500 or 1000×1000, subjected to frame dropping at regular intervals while retaining only 32 or 16 of the original frames and then interpolated to obtain 64 number of frames, followed by $5°$ or $8°$ rotation attacks respectively. The ROC curves for these multiple attacks are shown in Figure 2.25. In the first case, the performance of the video hashing algorithms based on the PARAFAC decomposition and the 3D-DCT is relatively stable, and the remaining video hashing algorithms considered show relatively poor performance. In the second case, the performance of all the video hashing algorithms has deteriorated. The video hashing algorithm based on the CGO shows poor performance in both the cases.

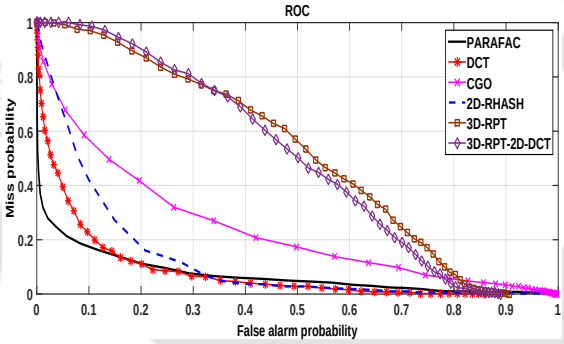
Decreased spatial resolution, frame-dropping and interpolation followed by average blurring attacks: In this attack, the spatial resolution of each frame was decreased to 250×250 or 100×100, subjected to frame dropping at regular intervals while retaining only 32 or 16 of the original frames and then interpolated to obtain 64 number of frames, followed by



(a) Adding zero mean AWGN with $\sigma = 57$ and Gaussian blurring using a 3×3 mask.

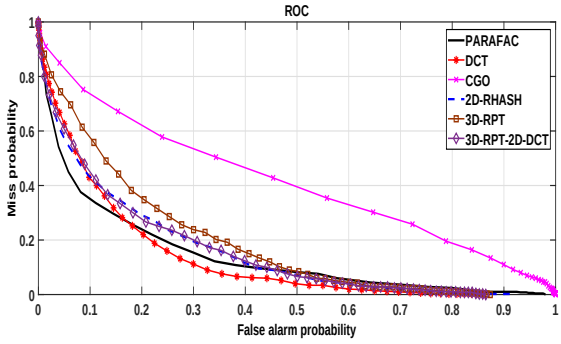


(b) Adding zero mean AWGN with $\sigma = 81$ and Gaussian blurring using a 5×5 mask.

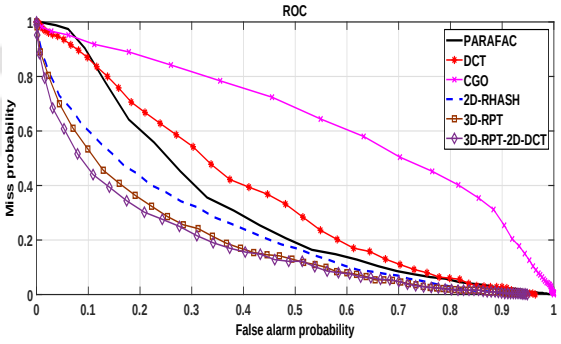


(c) Adding zero mean AWGN with $\sigma = 114$ and Gaussian blurring using a 9×9 mask.

Figure 2.23: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN and Gaussian blurring attacks.

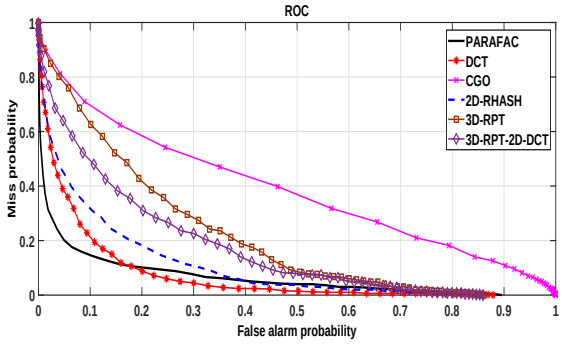


(a) Only 16 frames retained and 5° rotation.

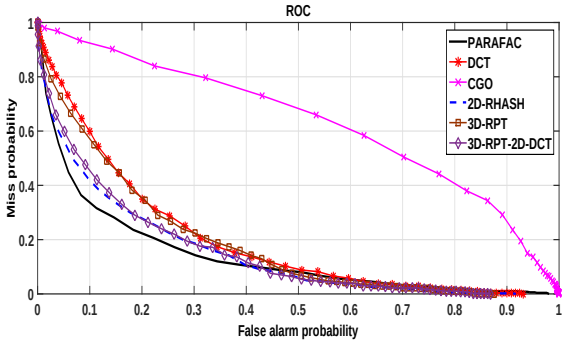


(b) Only 8 frames retained and 8° rotation.

Figure 2.24: Statistical evaluation of video hashing algorithms via the ROC curves under frame-dropping and interpolation followed by rotation attacks.

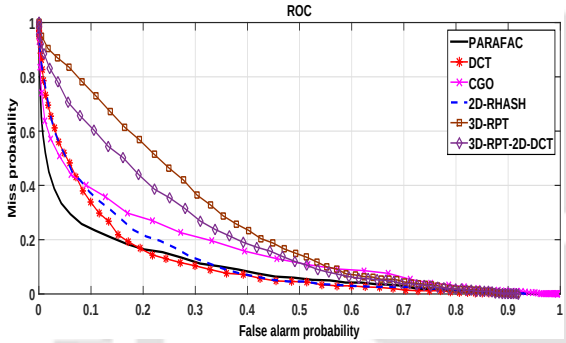


(a) Increase in spatial resolution to 500×500, only 32 frames retained and 5° rotation.

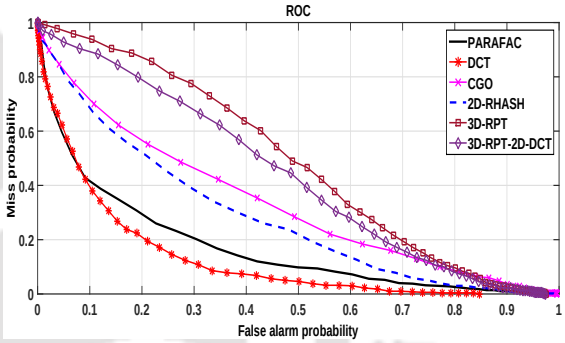


(b) Increase in spatial resolution to 1000×1000, only 16 frames retained and 8° rotation.

Figure 2.25: Statistical evaluation of video hashing algorithms via the ROC curves under increased spatial resolution, frame-dropping and interpolation followed by rotation attacks.



(a) Decrease in spatial resolution to 250×250, only 32 frames retained and average blurring using 5×5 mask.

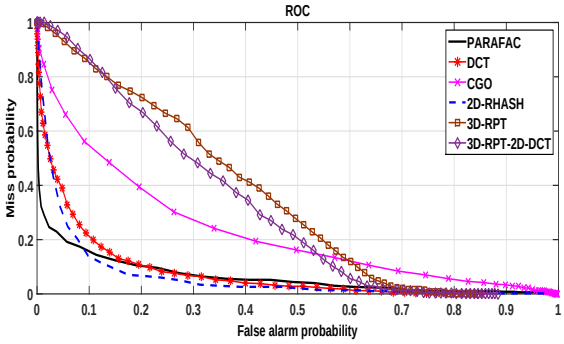


(b) Decrease in spatial resolution to 100×100, only 16 frames retained and average blurring using 9×9 mask.

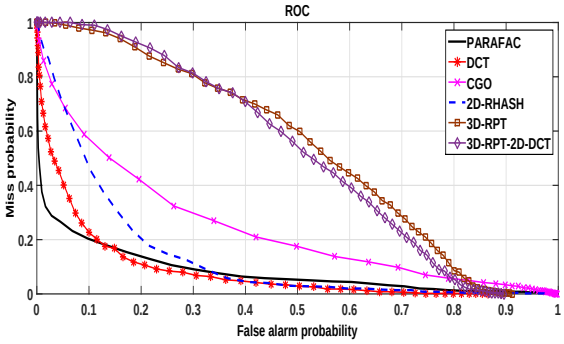
Figure 2.26: Statistical evaluation of video hashing algorithms via the ROC curves under decreased spatial resolution, frame-dropping and interpolation followed by average blurring attacks.

mean blurring using a mask of 5×5 or 9×9 respectively. The ROC curves for these multiple attacks are shown in Figure 2.26. In the first case, the performance of the video hashing algorithms based on the PARAFAC decomposition, the 3D-DCT and 2D-RASH is similarly good. The other video hashing algorithms considered show relatively poor performance. In the second case, the performance of the video hashing algorithms based on the PARAFAC decomposition and the 3D-DCT is relatively good while the other video hashing algorithms considered show relatively poor performance.

Addition of AWGN, frame insertion and Gaussian blurring attacks: In this attack, the video frames were subjected to zero mean AWGN with $\sigma = 81$ or $\sigma = 114$, insertion of 4 or 8 unwanted frames and the Gaussian filtering with a mask size of 5×5 or 9×9 respectively. The corresponding ROC curves are shown in Figure 2.27. This is a combination of content-preserving and the malicious attacks. The video hashing algorithms based on the PARAFAC decomposition, the 3D-DCT and the 2D-RASH are having a relatively smaller area under the ROC curves and fail to capture the malicious attack. While the video hashing algorithms based on the CGO, Scheme-I and Scheme-II of the 3D-RASH having a relatively larger area under



(a) Adding a zero mean AWGN with $\sigma = 81$, 4 unwanted frames inserted and Gaussian blurring using a 5×5 mask.



(b) Adding a zero mean AWGN with $\sigma = 114$, 8 unwanted frames inserted and Gaussian blurring using a 9×9 mask.

Figure 2.27: Statistical evaluation of video hashing algorithms via the ROC curves under addition of AWGN, frame insertion and Gaussian blurring attacks.

the ROC curves and hence capable of capturing the malicious attack.

2.5 Concluding Remarks

The performance of the proposed video hashing algorithm was found to be robust to the following single image processing attacks: rotation, cropping, modified brightness, frame dropping and interpolation, modified spatial resolutions, compression, reverse play, unwanted frame insertions and insertion of a logo as the watermark. The proposed hash is susceptible to the addition of the noise. For most of the multiple attacks, the performance of the proposed video hashing algorithm was found to be inferior compared to the other video hashing algorithms considered. From the validation and the performance evaluation of the hash, it can be concluded that the performance of Scheme-II of the radial projection based video hashing algorithm is better than Scheme-I of the 3D-RASH based video hashing algorithm for all the image processing attacks considered in our work. Furthermore, the proposed algorithm requires a lot of computations to extract the perceptual hash from a given video.

2.6 Summary of the Chapter

This chapter is summarized as follows.

- The perceptual hash was extracted from the variance of the luminance values projected along the line.
- The hash was generated using these feature vectors using two schemes.
  - In the first scheme, these feature vectors were averaged across the rows and then projected on to the 1D-DCT basis to generate a compact hash vector.
  - In the second scheme, the feature vectors were projected on to the 2D-DCT basis and then averaged to generate a compact hash vector.

- The proposed hash based on 3D-RASH was experimentally validated for its desirable properties. The characteristics of the obtained hash were away from the desired properties described in Section 1.1.
- The performance of the proposed method was evaluated using the ROC curves and compared with the video hashing algorithms based on the PARAFAC decomposition, the 3D DCT, the CGO, and the 2D RASH.
- From the ROC curves, it can be concluded that the performance of the algorithm was satisfactory. The main drawback of the algorithm is that the proposed hash is less robust to the addition of the noise and multiple attacks. The other drawback is that both the schemes require a lot of computations to generate the perceptual video hash.

Chapter 3

Perceptual Video Hashing based on Random Projection

Contents of the Chapter

The previous chapter proposed perceptual video hashing methods based on three-dimensional radial projections of the video in overlapping sub-blocks. The algorithm required a lot of computations to generate the perceptual hash from the video. The present chapter explores the application of computationally efficient random projection for developing perceptual video hashing methods.

Applying the dimensionality reduction technique through random projection such as *Fast Johnson-Lindenstrauss transform* (FJLT), Xudong and Wang in [16, 17], proposed an image hashing algorithm. The algorithm is robust to a large class of content-preserving distortions with less computational cost. It is observed that the results are comparable to the state-of-the-art image hashing algorithms such as image hashing based on the *Singular Value Decomposition*

(SVD) [2] and the *Non-negative Matrix Factorization* (NMF) [5]. The random projection techniques have not been applied for developing the perceptual hash for a video. We propose to apply such projections for video hashing.

The following advantages of the random projection technique motivate us to apply them for the generation of video hashes:

1. The projections of points from high dimensional data to a low dimension through a random projection technique approximately preserves the length of the unit vectors and also the pairwise distances among the points.

2. The high computational efficiency and security due to the random projection make it suitable for practical implementation.

3. The random projection technique can capture the essential features of the original data in a low dimensional subspace with minor distortion.

In our work, we propose to use the random projection matrix proposed by Dimitris Achlioptas [20, 21] for video hashing. The reason for choosing the Achlioptas's random matrix (ARM) over the other random projection matrix is discussed in Section 3.1. We also propose to apply the ARM on temporal low-pass frames of video rather than on the entire video.

The flow of the chapter is as per the following. The framework for the Achlioptas's random projection is described in Section 3.1. The perceptual video hashing algorithm using the ARM is proposed in Section 3.2. The validation and the performance evaluation of the ARM-based hash are presented in Section 3.3. The motivation for the video hashing algorithm based on the temporal wavelet transform and the Achlioptas's random matrix (TWT-ARM) is presented in Section 3.5. The framework for the temporal wavelet transform (TWT) is discussed in Section 3.6. The TWT-ARM based video hashing algorithm is proposed in Section 3.7. The validation of perceptual hash properties and the performance evaluation of the random projection based video hashing algorithm is presented in Section 3.8.

3.1 Dimensionality reduction using random projections

Random projections have emerged as a powerful tool for dimensionality reduction. The results obtained through the random projection are comparable to that of the conventional dimensionality reduction techniques such as the Principal Component Analysis (PCA) and the Non-negative Matrix Factorization (NMF). Random projections are computationally less expensive than the conventional dimensionality reduction methods [38]. The applications of random projection include searching of approximate nearest neighbours in high-dimensional Euclidean space [39, 22, 23], dimension reduction in databases [40] and learning high-dimensional Gaussian mixture models [41].

When the original high-dimensional data are projected onto a low-dimensional space using a random projection matrix, the distance between the vectors are preserved, provided the projection or the mapping satisfies the *Johnson-Lindenstrauss* (JL) lemma [32]. Johnson and Lindenstrauss proved that any subset of n points in the d-dimensional Euclidean space can be

embedded into the k-dimensional Euclidean space, where $k = \mathcal{O}\left(\epsilon^{-2} \ln n\right)$, without distorting the distances between any pair of points by more than a factor of $(1 \pm \epsilon)$, for any $0 < \epsilon < 1$. The dimension k is independent of d. The JL lemma is given by:

Johnson-Lindenstrauss Lemma*: For any $0 < \epsilon < 1$ and any integer n, let k, be a positive integer such that $k \geq \frac{4 \times \ln n}{\left(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}\right)}$. Then, for any subset $\mathbf{Q}$ of n points in $\mathbb{R}^d$, there is a map $f : \mathbb{R}^d \to \mathbb{R}^k$ such that for all $\mathbf{u}, \mathbf{v} \in \mathbf{Q}$,*

$$(1 - \epsilon) \left\|\mathbf{u} - \mathbf{v}\right\|_2^2 \leq \left\|f\left(\mathbf{u}\right) - f(\mathbf{v})\right\|_2^2 \leq (1 + \epsilon) \left\|\mathbf{u} - \mathbf{v}\right\|_2^2 \tag{3.1}$$

Further this map can be found in a randomized polynomial time.

The proof for the JL lemma is given in [40, 32]. The concept of randomized polynomial time can be found in [42, 43, 44]. The mapping of data using the JL lemma requires a dense $k \times d$ matrix. So mapping each point takes $\mathcal{O}\left(d \ln n\right)$ time for fixed ϵ [22, 23].

Dimitris Achlioptas proposed a simple probability distribution to generate the projection matrix in [20, 21] that satisfies the JL lemma. Each element of this matrix is either 1 or -1, and thus the projection involves only additions and subtractions. We call this projection matrix as *Achlioptas's Random Matrix* (ARM). Achlioptas's method for random projection is summarized in a theorem in [20, 21]. The same is reproduced here.

Theorem 1 (Achlioptas's theorem) *Let $\mathbf{Q}$ be an arbitrary subset of n points in $\mathbb{R}^d$, represented as an $d \times n$ matrix $\mathbf{F}$. Given $\epsilon, \beta > 0$, let $k_0 = \frac{(4+2\beta) \times \ln n}{\left(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}\right)}$. For any integer $k \geq k_0$, suppose $\mathbf{R}_{ARM}$ is a $k \times d$ random matrix and the elements $\mathbf{R}_{ARM}\left(i, j\right)$, $i = 1, 2, \ldots, k$; $j = 1, 2, \ldots, d$, are independent and identically distributed random variables following the probability distribution:*

$$\mathbf{R}_{ARM}\left(i, j\right) = \begin{cases} +1 & \textit{with probability 0.5} \\ -1 & \textit{with probability 0.5} \end{cases}. \tag{3.2}$$

Let $\mathbf{E} = \frac{1}{\sqrt{k}} \mathbf{R}_{ARM} \mathbf{F}$ and $f : \mathbb{R}^d \to \mathbb{R}^k$ map the i^{th} column of $\mathbf{F}$ to the i^{th} column of $\mathbf{E}$, then with probability at least $1 - n^{-\beta}$ and $\forall\, \mathbf{u}, \mathbf{v} \in \mathbf{Q}$,

$$(1 - \epsilon) \left\|\mathbf{u} - \mathbf{v}\right\|_2^2 \leq \left\|f\left(\mathbf{u}\right) - f(\mathbf{v})\right\|_2^2 \leq (1 + \epsilon) \left\|\mathbf{u} - \mathbf{v}\right\|_2^2$$

The proof for the above theorem can be found in [20, 21].

The reasons for choosing the ARM as the projection matrix over the other random projection matrices such as Gaussian and FJLT are enumerated as follows:

1. the ARM comprises either 1 or -1, and thus, the projection involves only additions and subtractions
2. the ARM projection matrix satisfies the JL lemma and hence, retains most of the relevant features of the high dimensional data in the low dimensional subspace

The focus of this work is on developing a computationally efficient, and simple-to-implement algorithm and this is achieved by the ARM-based projection. Furthermore, the Gaussian or the FJLT random projection matrices involve real multiplication and are computationally more complex.

3.2 Perceptual Video Hashing using ARM

The generic block diagram of the proposed video hashing algorithm based on the random projection is shown in Figure 3.1. The algorithm consists of the following blocks: *preprocessing and normalization*, *randomization*, *random projections* and *hash computation*. The algorithm generates the hash by capturing the spatiotemporal essence of the input video and hence belongs to the class of spatiotemporal based video hashing algorithms. The brief description of the blocks is as follows:

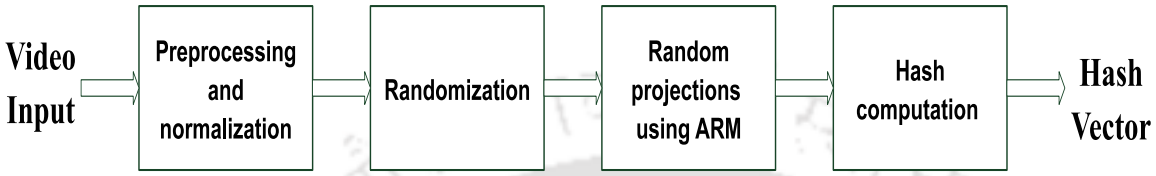


Figure 3.1: The block diagram representation for the extraction of perceptual video hash using the Achlioptas's random projection.

(i) *Preprocessing and Normalization*: The input videos might have the variations in their spatial dimensions and the number of frames. Since the resultant hash size of the algorithm is kept constant, standardize the spatial size and the number of frames in the videos to $X \times X \times X$ via temporal sub-sampling and spatial re-sizing. This normalization process ensures the impact of various spatial dimensions and frame rates on the hash value to be minimal. Let the normalized video be represented by $\mathcal{V}$.

(ii) *Randomization*: Depending on a secret key K, randomly select N overlapping 3D sub-blocks of video of pre-defined size $U \times U \times U$, such that $U < X$. Let $\mathcal{V}_i$, $i = 1, 2, \ldots, N$, be the randomly overlapping 3D sub-blocks. The secret key ensures the security in the hash generation process. The number of overlapping 3D sub-blocks of the video is chosen using trial and error, approximately covering the entire normalized video $\mathcal{V}$. As the selection of the key is randomized, the selection of overlapping 3D sub-blocks is also different, and hence the resultant hash is also distinct. Each 3D sub-block is rearranged to form a column of the feature matrix, say $\mathbf{F} = [\mathbf{f}_1\ \mathbf{f}_2\ \ldots\ \mathbf{f}_N]$. $\mathbf{F}$ captures the local information of the normalized video in each column and the global information as a whole. The advantage of $\mathbf{F}$ is that it is robust under geometric attacks [23]. Because, even if the portions of the original frames are lost under geometric attacks such as cropping, it will only affect one or a few components in the feature matrix and have no significant influence on the global information.

(iii) *Random Projections*: Each column vector of $\mathbf{F}$ is of high dimension. Use the random projection to reduce the dimension. Map the high dimensional data into the low dimensional data using the random projection method, namely the ARM. As discussed in the earlier section, the mapping of the data via the ARM technique using the random projection matrices, $\mathbf{R}_{ARM}$, satisfies the JL lemma. Thus the random projection technique is able

to capture the essential features of the original data in a low dimensional subspace with minor distortion if the factor ϵ is close to 0. The low dimensional data is enclosed in the hash matrix

$$\mathbf{H} = \frac{1}{\sqrt{k}}\mathbf{R}_{ARM}\mathbf{F} \tag{3.3}$$

where k represents the dimension of the reduced sub-space which is also the length of the hash.

(iv) *Hash Computation*: The arithmetic averaging operation is performed on the low dimensional data to obtain the intermediate hash vector,

$$\mathbf{h}' = \frac{\sum_{i=1}^{N}\mathbf{h}_i}{N}, \tag{3.4}$$

where $\mathbf{h}_i$s are the columns of the matrix $\mathbf{H}$. This averaging operation not only preserves the robustness and the fragility of the hash but also reduces the length of the hash suitable for practical applications. Finally, the video hash $\mathbf{h}$ is obtained by binarizing the intermediate hash vector elements using the median ξ, of the *intermediate hash* vector elements as the threshold. Mathematically,

$$h_j = \begin{cases} 0; & h'_j < \xi \\ 1; & h'_j \geq \xi \end{cases}, \; j = 1, 2, \ldots, k. \tag{3.5}$$

where h'_j and h_j are the jth elements of $\mathbf{h}'$ and $\mathbf{h}$, respectively.

The algorithmic steps of the proposed hashing method are summarized in the Algorithm 3.1

3.3 Simulation Results and Discussion

A number of experiments are conducted to validate the hash properties and evaluate the performance of the hash. The details of the database are tabulated in Table 2.1 and discussed in Chapter 2. In this set of experiments, all the videos are normalized to $64 \times 64 \times 64$. 32 randomly overlapping sub-cubes of size $16 \times 16 \times 16$ are chosen using a secret key such that they cover the entire video. Thus each feature vector is of size 4096. The number 32, is selected by trial and error to get an optimal performance of the algorithm. To have a fair comparison with other algorithms, a uniform hash length of 128 is considered. Accordingly, the values of ϵ and β in Achlioptas's theorem were chosen to be 0.725 and 0.5 respectively.

3.3.1 Hash Validation

Experiments were conducted to validate the unpredictability, the visual fragility and the perceptual robustness properties of the hash generated using the ARM-based video hashing algorithm. 224 videos from the database and 1000 secret keys are considered to validate the proposed hash. The NHD was the distance metric used to compare the hash values of the videos. The histograms of the NHD to evaluate the important desirable properties of the hash are shown in Figure 3.2. The validations are described below.

Algorithm 3.1 *Proposed video hashing algorithm based on the ARM.*

1: **Input**

- RGB video V_{in} and k (even).
- Algorithm parameters: X, U, N, K, ϵ and β.

2: **Preprocessing and normalization**

- Convert V_{in} into a sequence of grey-level frames, V_{grey}.
- Normalize V_{grey} via temporal sub-sampling and spatial re-sizing to obtain $\mathcal{V}$ of size $X \times X \times X$.

3: **Randomization**

- Depending on secret key K, randomly select N overlapping 3D sub-blocks $\mathcal{V}_i$, $i = 1, 2, \ldots, N$, each of size $U \times U \times U$, such that $U < X$.
- Concatenate the columns of each sub-block $\mathcal{V}_i$ to form vector $f_i \in \mathbb{R}^d$ such that $d = U \times U \times U$.
- Construct the feature matrix $\mathbf{F} \in \mathbb{R}^{d \times N}$ from f_is to obtain $\mathbf{F} = [\mathbf{f}_1\ \mathbf{f}_2\ \ldots\ \mathbf{f}_N]$.

4: **Random projections**

- Apply the dimensionality reduction technique to $\mathbf{F}$ using
  - the ARM projection matrix $\mathbf{R}_{ARM} \in \mathbb{R}^{k \times d}$ to get the hash matrix $\mathbf{H} = \frac{1}{\sqrt{k}}\mathbf{R}_{ARM}\mathbf{F}$.

5: **Hash computation**

- Generate the intermediate hash vector $\mathbf{h}' = \frac{\sum_{i=1}^{N} \mathbf{h}_i}{N}$, where $\mathbf{h}_i$ is the i^{th} column of the matrix $\mathbf{H}$.
- Rank order $\mathbf{h}'$ to obtain $\mathbf{h}'' = [h'_{(1)}\ h'_{(2)} \ldots h'_{(k)}]^T$.
- Median, $\xi = \left(\frac{h'\left(\frac{k}{2}\right) + h'\left(\frac{k}{2}\right)+1}{2} \right)$.
- Binarize the elements of $\mathbf{h}'$ to obtain $\mathbf{h} = [h_1\ h_2 \ldots h_k]^T$ where

$$h_j = \begin{cases} 0; & h'_j < \xi \\ 1; & h'_j \geq \xi \end{cases},\ j = 1, 2, \ldots, k.$$

6: **Output**

- A hash vector $\mathbf{h}$

(i) *To validate the unpredictability property*

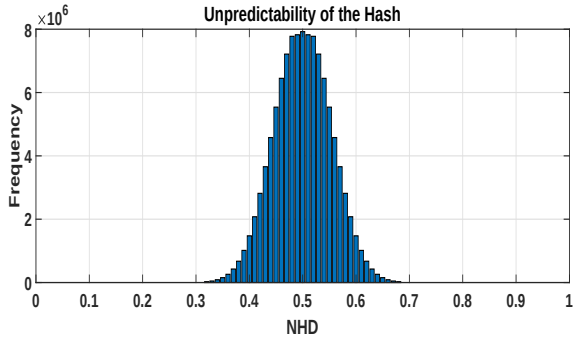
128-bit hashes were generated for a test sequence using 1000 different secret keys, and then, the NHD was measured between every possible pair of the hashes. The procedure was repeated for the remaining 223 test sequences. The histogram of the NHD values for this experiment is shown in Figure 3.2(a). It can be observed from this figure, that the NHD measured between the hashes approximately follows a Gaussian distribution

with the mean 0.5 and standard deviation of 0.106. A narrow-spread distribution centred around 0.5 implies that the perceptual hash closely satisfies the unpredictability property given in Equation 1.5. Comparing Figure 3.2(a) and Figure 2.5(a), we can conclude that the perceptual hash based on ARM satisfies the unpredictability property better than the perceptual hash based on the 3D-RASH.

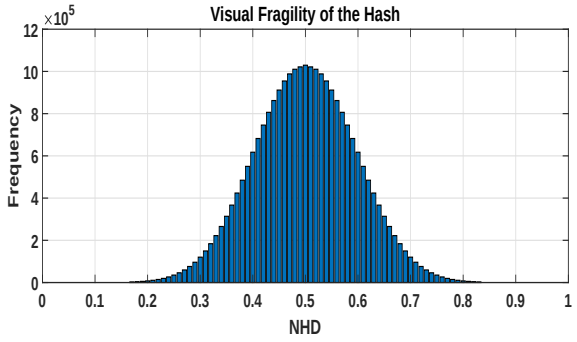
(ii) *To validate the visual fragility property*
128-bit hashes were generated for 224 test sequences using the same secret key, and then, the NHD was measured between every possible pair of the hashes. The procedure was repeated for the remaining 999 secret keys. The histogram of the NHD values for this experiment is shown in Figure 3.2(b). It can be observed from this figure, that the NHD measured between the hashes approximately follows Gaussian distribution with the mean 0.5 and standard deviation of 0.193. A narrow-spread distribution centred around 0.5 implies that the perceptual hash closely satisfies the visually fragile property given in Equation 1.4. Comparing Figure 3.2(b) and Figure 2.5(b), we can conclude that the perceptual hash based on ARM satisfies the visual fragility property better than the perceptual hash based on the 3D-RASH.

(iii) *To validate the perceptual robustness property*
All the videos were corrupted with AWGN of $\sigma = 81$ and subsequently blurred with the Gaussian mask of size 5×5 thereby generating a set of perceptually similar videos. The hashes were generated for these videos using the same secret key. The NHD between the hashes of the original videos and the perceptually similar videos was measured. A histogram skewed towards zero implies that the perceptual hash closely satisfies the perceptual robustness property given in Equation 1.3. The histogram of all the NHD values for this experiment is shown in Figure 3.2(c) and we observe that the NHD between the original videos and the perceptually similar videos are distributed between 0 and approximately 0.5 with the mean centred around 0.25. Therefore, we conclude that the perceptual hash based on ARM satisfies the perceptual robustness property better than the perceptual hash based on the 3D-RASH.

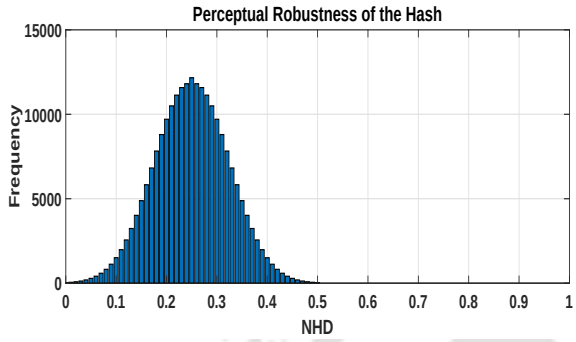
(iv) *To assess perceptual robustness versus visual fragility*
To evaluate the results of perceptual similarity obtained from the ARM-based video hashing algorithm for the perceptually similar videos and distinct videos, the histograms of the NHDs obtained for the perceptual robustness property and the visual fragility property are superimposed as shown in Figure 3.2(d). From the figure, it can be observed that the histogram of the NHD for the perceptually similar videos and distinct videos overlap. The overlapping indicates that there would be misclassification between the perceptually similar and distinct videos. From Figure 3.2(d), it can be observed that there is overlapping between the histograms and hence, there would be misclassification. However, the overlapping here is less compared to Figure 2.5(d). Hence, the performance of the video hashing algorithm based on the ARM is better than the video hashing algorithm based on the 3D-RASH.



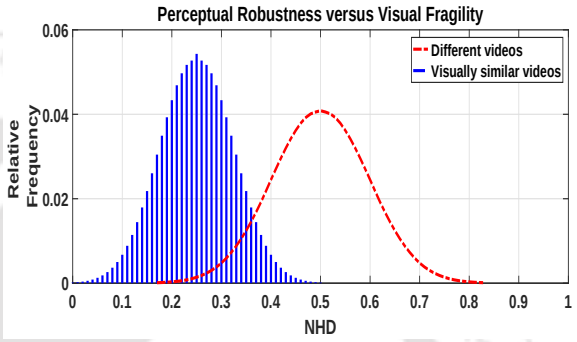
(a) Unpredictability property of the hash.



(b) Visual fragility property of the hash.



(c) Perceptual robustness property of the hash.



(d) Perceptual robustness property versus the visual fragility of the hash.

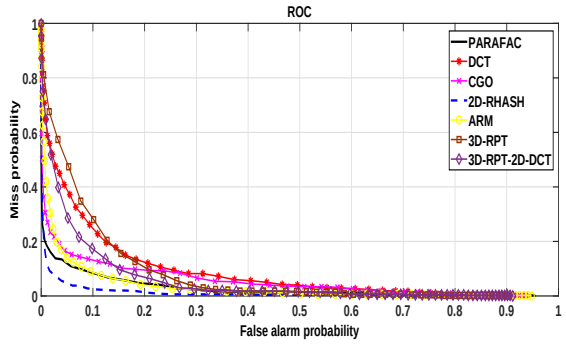
Figure 3.2: The histogram of normalized Hamming distance to evaluate the important desirable properties of the hash generated using the ARM-based video hashing algorithm.

From Figure 3.2, we conclude that the perceptual hash generated from the proposed algorithm closely satisfies all the necessary property of the perceptual hash.

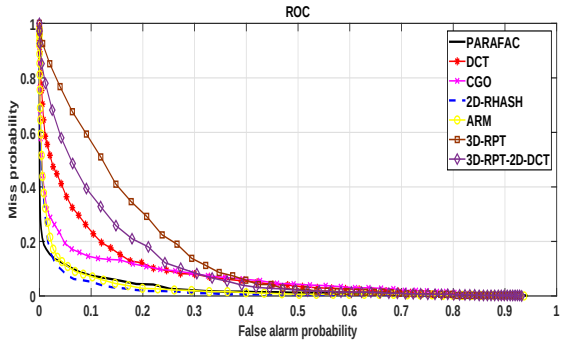
3.3.2 Evaluation of Hash Performance

The ROC curve is used to evaluate the performance of the video hashing algorithms. The ROC curve shows the miss probabilities at various false-alarm probabilities. The following perceptual video hashing algorithms were considered for comparison with the ARM-based video hashing algorithm.

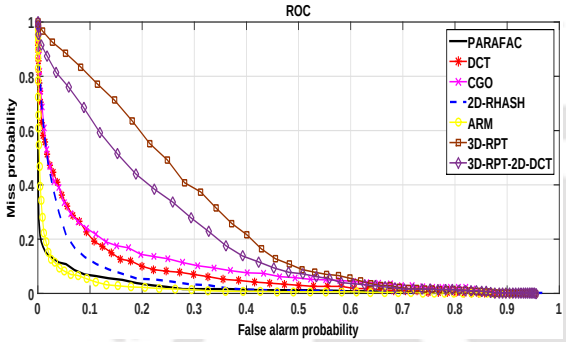
- the PARAFAC decomposition based video hashing algorithm [15], labelled as PARAFAC in the ROC curve.
- the DCT based video hashing algorithm [12, 13], labelled as DCT in the ROC curve.
- the CGO based video hashing algorithm [7, 27], labelled as CGO in the ROC curve.
- the 2D radial projection based video hashing algorithm [11, 10], labelled as 2D-RHASH in the ROC curve.
- the video hashing algorithm based on the 3D-RASH using Scheme-I and Scheme-II are respectively labelled as 3D-RPT and 3D-RPT-2D-DCT in the ROC curve.



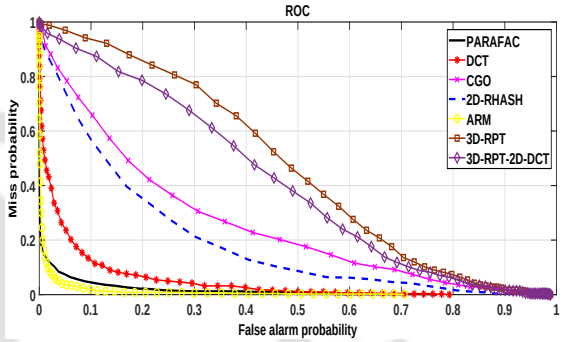
(a) 5×5.



(b) 9×9.



(c) 15×15.



(d) 35×35.

Figure 3.3: Statistical evaluation of video hashing algorithms via the ROC curves under average blurring attack using different mask size.

The proposed ARM-based video hashing algorithm, labelled as ARM in the ROC curve. The same single/multiple image processing attacks as in Chapter 2 are considered here.

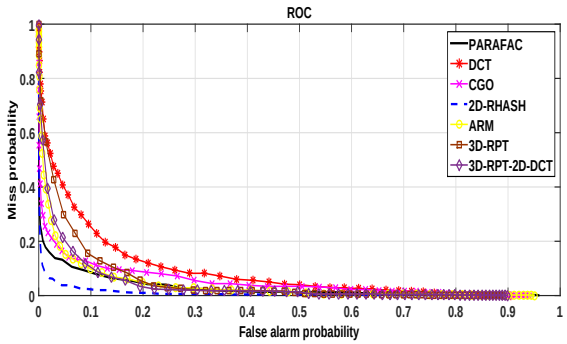
3.3.2.1 Single Attacks

The evaluation of hash performance for various single attacks is as follows. The attacks include the image processing and the malicious attacks described earlier.

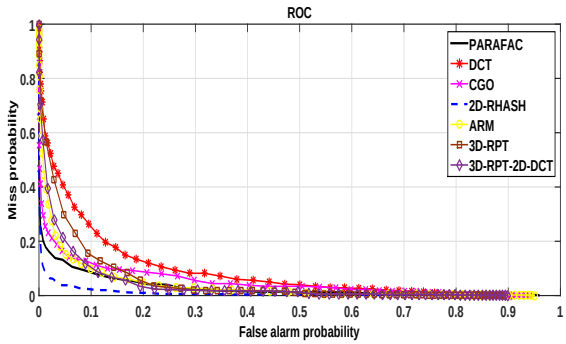
Average blurring attack: Figure 3.3 shows the ROC curves under average blurring using the mask of the following sizes: 5×5, 9×9, 15×15 and 35×35. The performance of the video hashing algorithms based on the PARAFAC decomposition, the ARM and the 3D-DCT is relatively stable. The performance of the video hashing algorithm based on the 2D-RASH, the CGO, Scheme-I and Scheme-II of the 3D-RASH deteriorate for 15×15 and 35×35 mask sizes.

Gaussian blurring attack: The ROC curves under the Gaussian filtering using the 5×5, 9×9, 15×15 and 35×35 masks are shown in Figure 3.4. The performance of all the video hashing algorithms under considerations is similarly good for all the mask sizes. It can be observed that the performance of all the video hashing algorithms considered deteriorates as the mask size is increased to 15×15 and 35×35.

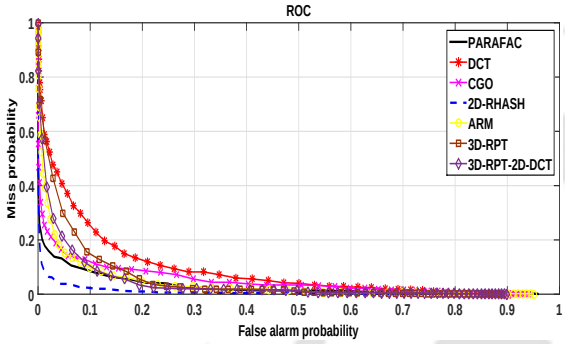
AWGN attack: Figure 3.5 shows the ROC curves for the zero mean AWGN attack with the following values of σ: 57, 81, 114 and 127. The performance of the video hashing algorithms based on the ARM, the PARAFAC decomposition and the 3D-DCT is relatively stable. As the



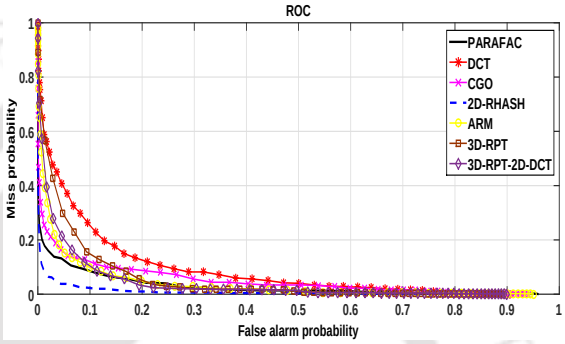
(a) 5×5.



(b) 9×9.

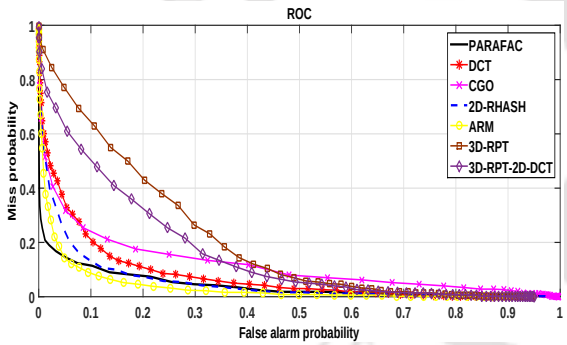


(c) 15×15.

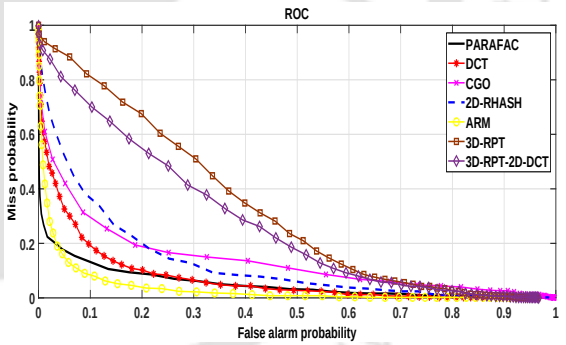


(d) 35×35.

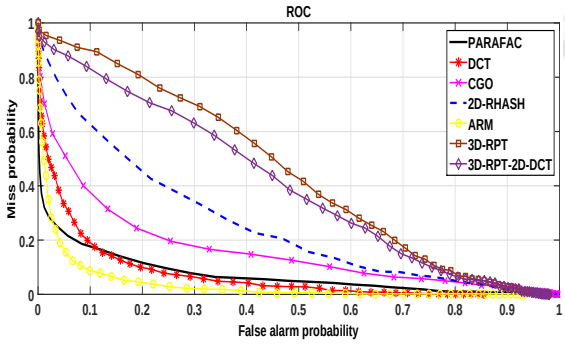
Figure 3.4: Statistical evaluation of video hashing algorithms via the ROC curves under Gaussian blurring attack using different mask size.



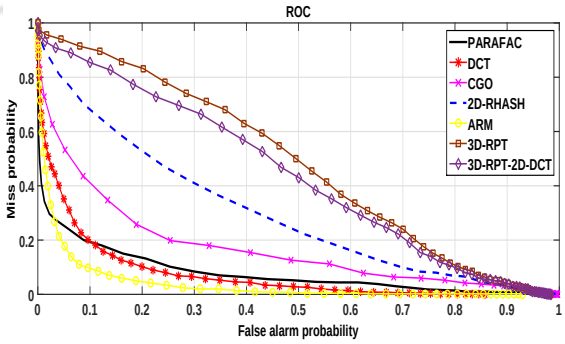
(a) Zero mean AWGN with $\sigma = 57$.



(b) Zero mean AWGN with $\sigma = 81$.

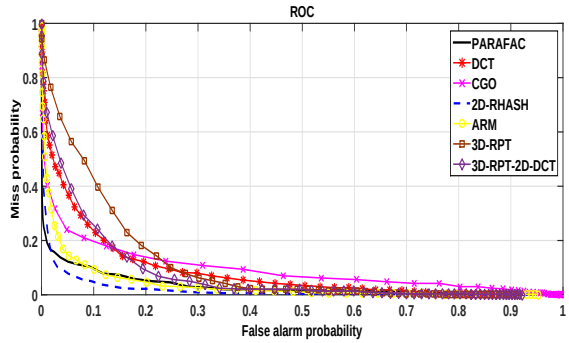


(c) Zero mean AWGN with $\sigma = 114$.

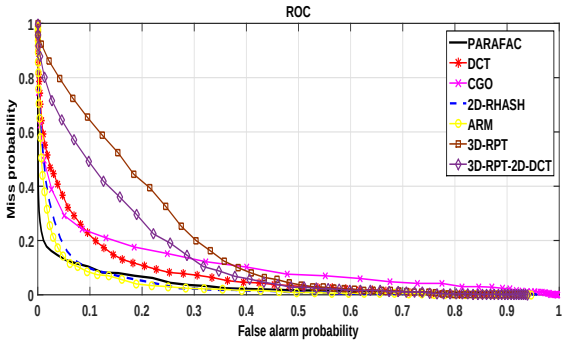


(d) Zero mean AWGN with $\sigma = 127$.

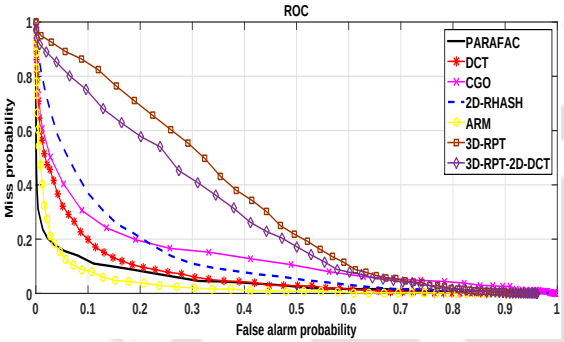
Figure 3.5: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN attack.



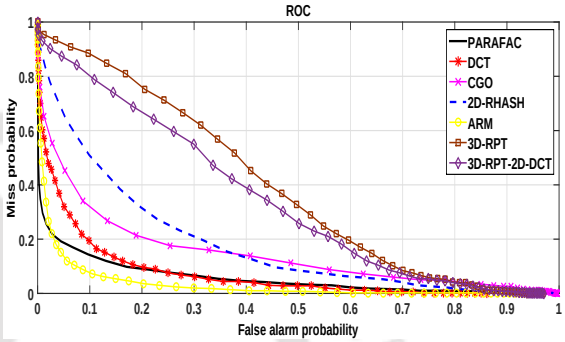
(a) Salt and pepper noise with 5% noise density.



(b) Salt and pepper noise with 10% noise density.



(c) Salt and pepper noise with 20% noise density.



(d) Salt and pepper noise with 25% noise density.

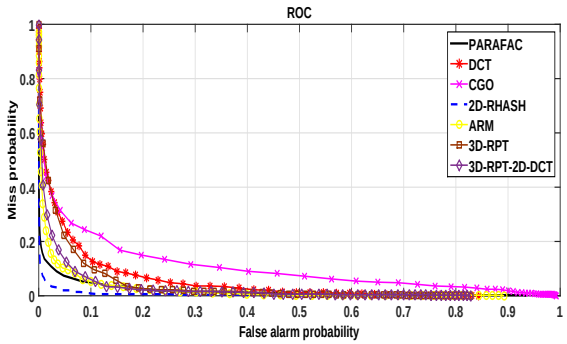
Figure 3.6: Statistical evaluation of video hashing algorithms via the ROC curves under salt and pepper noise attack.

value of σ increases, the performance of the video hashing algorithms based on the 2D-RASH, Scheme-I and Scheme-II of the 3D-RASH deteriorates quickly.

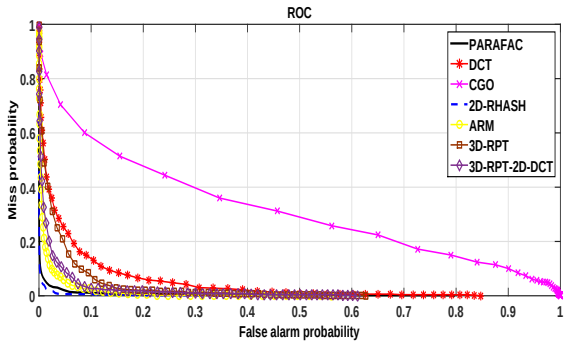
Salt and pepper noise attack: Figure 3.6 shows the ROC curves under the salt and pepper noise attack with the following noise densities: 5%, 10%, 20% and 25%. The performance of the video hashing algorithms based on the ARM, the PARAFAC decomposition and the 3D-DCT is relatively stable. It can be observed that as the density of the noise increases, the performance of the video hashing algorithm based on the 2D-RASH, Scheme-I and Scheme-II of 3D-RASH degrades quickly.

Rotation attack: Figure 3.7 shows the ROC curves under the rotation attack with the following degrees of rotation: 2°, 5°, 8° and 12°. The performance of all the video hashing algorithms considered is nearly the same for 2° and 5° rotation attack. For 8° and 12° rotation attack, the 3D-DCT based video hashing algorithm show poor performance while the CGO based video hashing algorithm shows least robustness to the rotation attack beyond 2°. The performance of the video hashing algorithm based on Scheme-I and Scheme-II of 3D-RASH is relatively stable for all the degree of rotation attack considered.

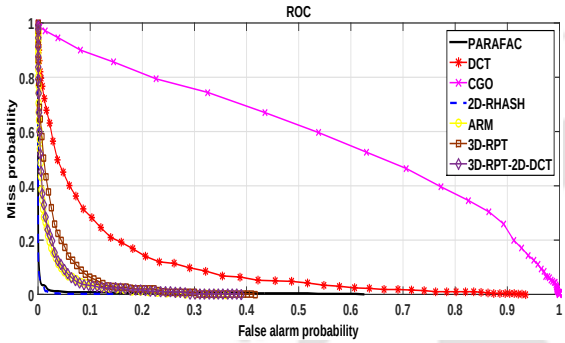
Cropping attack: The ROC curves for the cropping attack are shown in Figure 3.8. The following percentages of pixels were cropped from the borders of each frame: 5%, 10% and 15%. The performance of all the video hashing algorithms considered is similarly good except the 3D-DCT based video hashing algorithm. It can be observed from the figure that as more and more boundary pixels are cropped, the performance of the video hashing algorithm based on



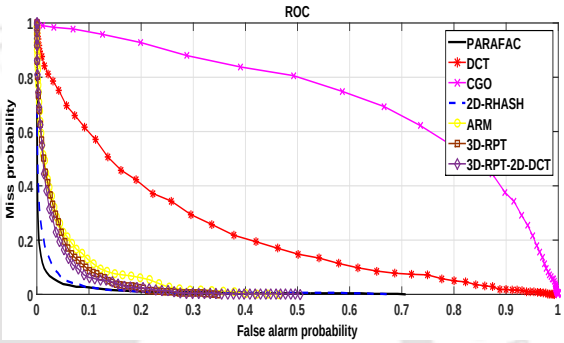
(a) 2°.



(b) 5°.

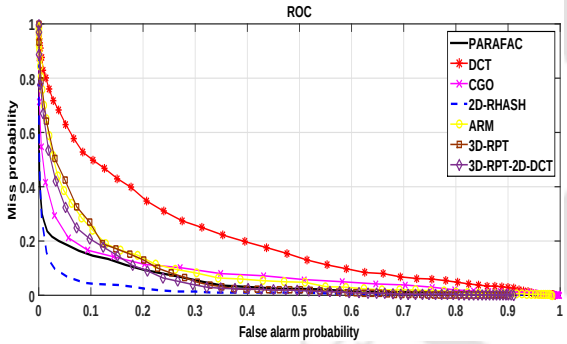


(c) 8°.

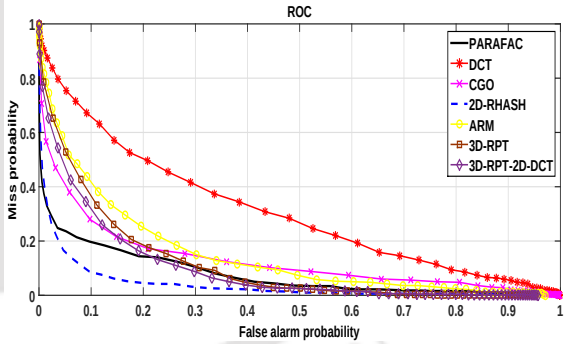


(d) 12°.

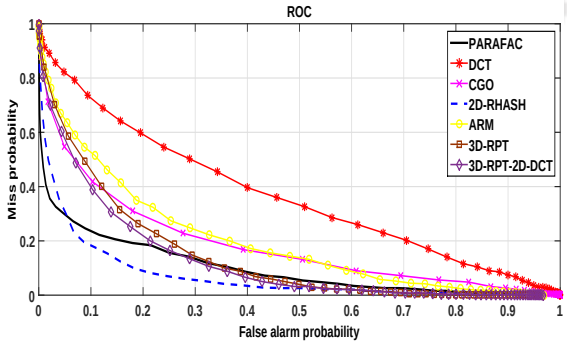
Figure 3.7: Statistical evaluation of video hashing algorithms via the ROC curves under rotation attack with different degrees of frame rotation.



(a) 5% crop.

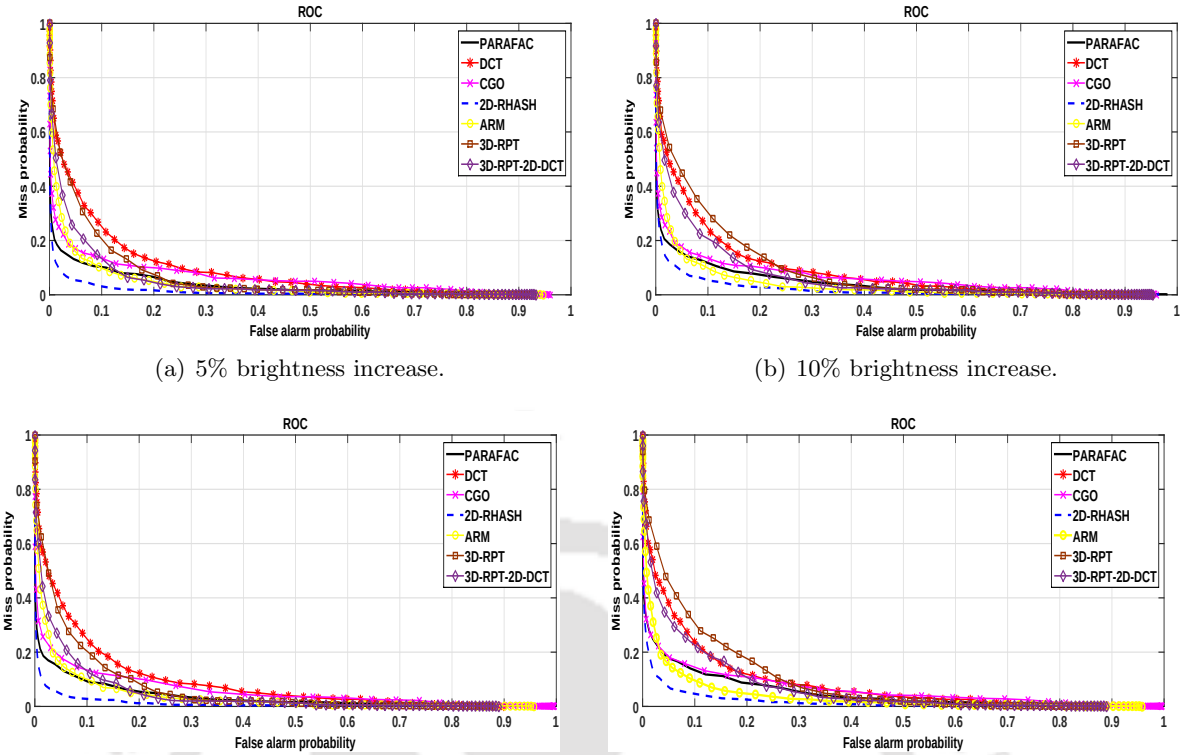


(b) 10% crop.



(c) 15% crop.

Figure 3.8: Statistical evaluation of video hashing algorithms via the ROC curves under cropping attack with different percentages of crop.



(a) 5% brightness increase.

(b) 10% brightness increase.

(c) 5% brightness decrease.

(d) 10% brightness decrease.

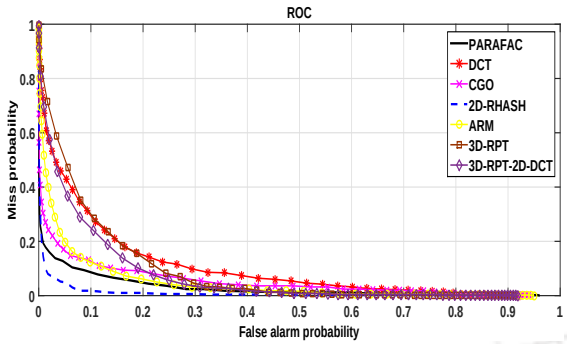
Figure 3.9: Statistical evaluation of video hashing algorithms via the ROC curves under brightness variation attack.

the 3D-DCT becomes progressively poor compared to the other video hashing algorithms. It can be observed that the 2D-RASH based video hashing algorithm followed by the other video hashing algorithms considered are relatively stable for the cropping attack.

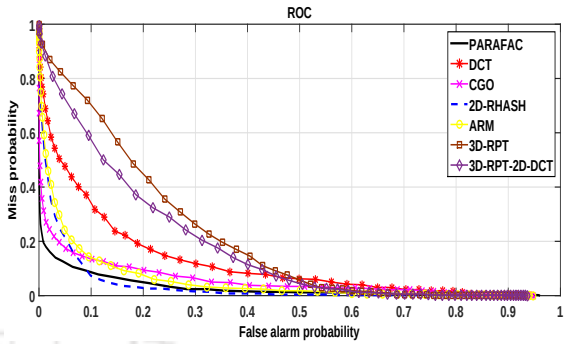
Brightness variation attack: The ROC curves under the modified brightness attack ($\pm 5\%$ and $\pm 10\%$) are shown in Figure 3.9. The performance of the 2D-RASH based video hashing algorithm is relatively stable. The performance of all the other video hashing algorithms considered is similar and deteriorates for $\pm 10\%$ modification in the brightness.

Contrast variation attack: The ROC curves for the modifications in the contrast are shown in Figure 3.10. The contrast of the frames is increased by multiplying the frame pixels with 1.1 and 1.2, and decreased by multiplying the frame pixels with 0.9 and 0.8, respectively. The performance of the PARAFAC decomposition based video hashing algorithm is relatively stable for the modified contrast attack followed by the video hashing algorithms based on the CGO, the 3D-DCT and the ARM. The performance of the other video hashing algorithms considered is robust to the increased contrast attack while susceptible to the decreased contrast attack. The video hashing algorithms based on the 2D-RASH, Scheme-I and Scheme-II of 3D-RASH are least robust to the contrast decrease attack.

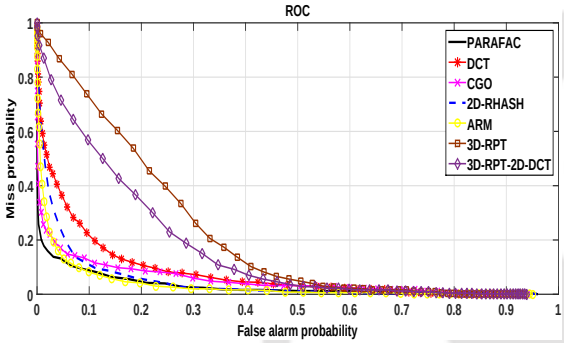
Frame-drop and interpolation attack: The ROC curves for the frame-drop and interpolation attack are shown in Figure 3.11. In the first case, only 48 frames were retained, and then the remaining 16 frames were interpolated using temporal averaging. It can be observed that the video hashing algorithms based on the 2D-RASH, the CGO and the PARAFAC decom-



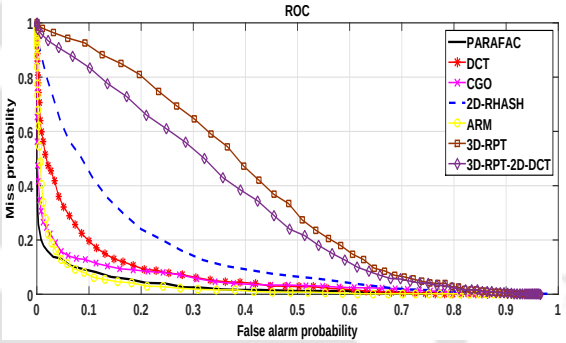
(a) 10% contrast increase.



(b) 20% contrast increase.

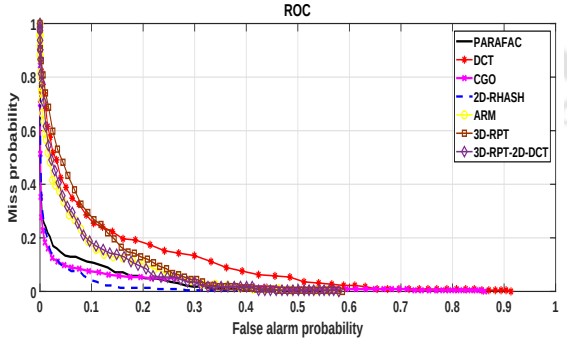


(c) 10% contrast decrease.

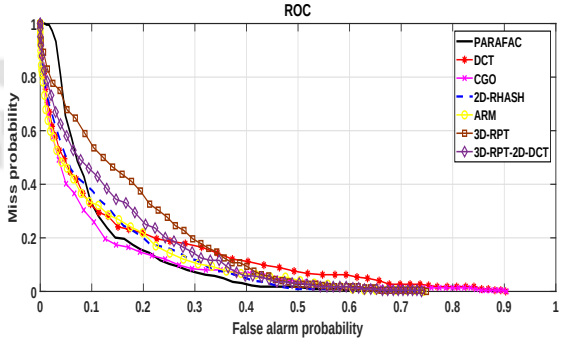


(d) 20% contrast decrease.

Figure 3.10: Statistical evaluation of video hashing algorithms via the ROC curves under contrast variation attack.

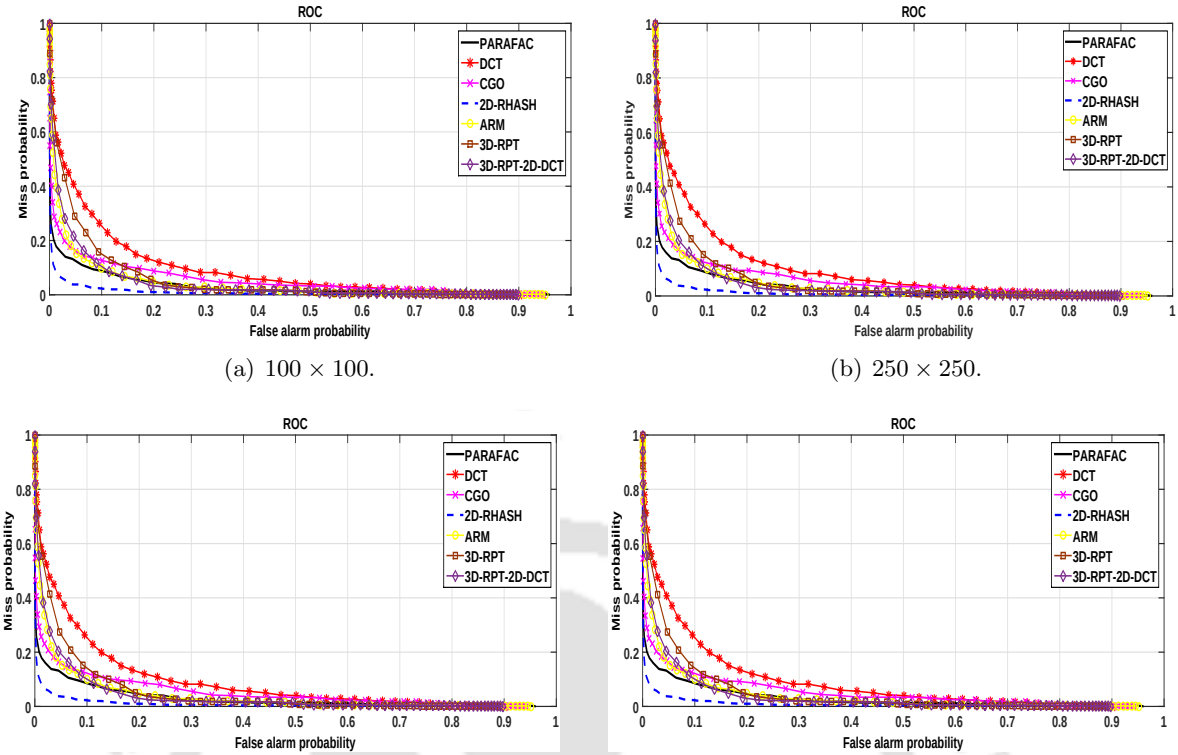


(a) 48 frames retained.



(b) 32 frames retained.

Figure 3.11: Statistical evaluation of video hashing algorithms via the ROC curves under frame-drop and interpolation attack.



(a) 100×100. (b) 250×250.

(c) 500×500. (d) 1000×1000.

Figure 3.12: Statistical evaluation of video hashing algorithms via the ROC curves under spatial resolution variation attack.

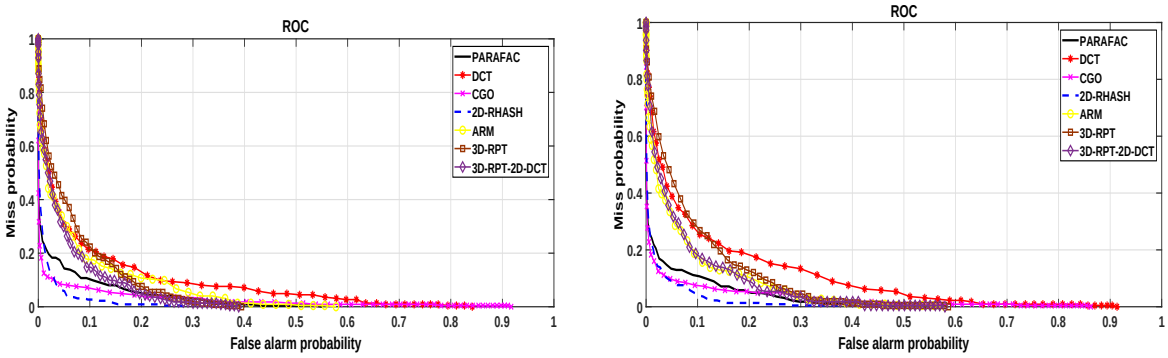
position show similar performance and has the least area under the ROC curves followed by the video hashing algorithm based on the ARM, Scheme-I and Scheme-II of 3D-RASH and the 2D-DCT. In the second case, only 32 frames were retained, and then the remaining 32 frames were interpolated using temporal averaging. In this case, all the video hashing algorithms considered show similar performance.

Spatial resolution variation: Figure 3.12 shows the ROC curves under the modified spatial resolutions attack. The following spatial resolutions were considered: 100×100, 250×250, 500×500 and 1000×1000. It can be observed that all the algorithms under considerations show similar performance. The 2D-RASH based video hashing algorithm is more robust to the variations in the spatial resolution while the 3D-DCT based video hashing algorithm is less robust to this attack.

Compression attack: Two video database was constructed with the average CR of 250:1 and 240:1, respectively. The statistical evaluation of video hashing algorithms using the ROC curves under the compression attack is shown in Figure 3.13. It can be observed that all the algorithms under consideration show similar performance.

Reverse play attack: The ROC curves under the reverse play attack are shown in Figure 3.14. The video hashing algorithm based on Scheme-I and Scheme-II of 3D-RASH is robust to this attack. The ROC curves for the remaining video hashing algorithms stayed close to the line of no discrimination implying the poor performance of all the algorithms considered.

Frame insertion attack: Firstly, four frames of a different video are inserted in the



(a) 100kbps bit-rate and 240:1 average CR.

(b) 64kbps bit-rate and 250:1 average CR.

Figure 3.13: Statistical evaluation of video hashing algorithms via the ROC curves under compression attack.

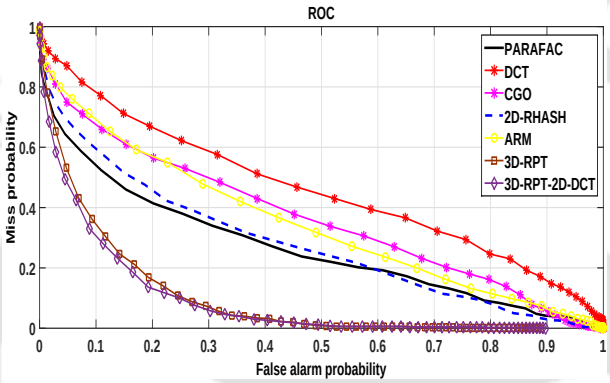
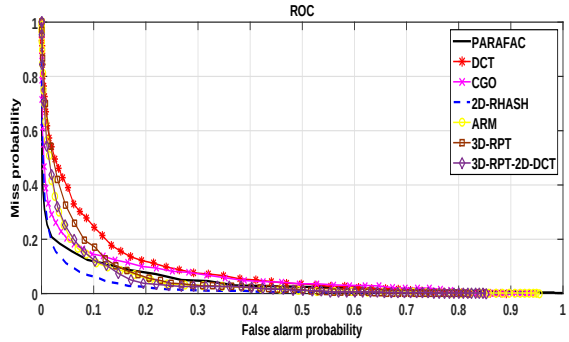


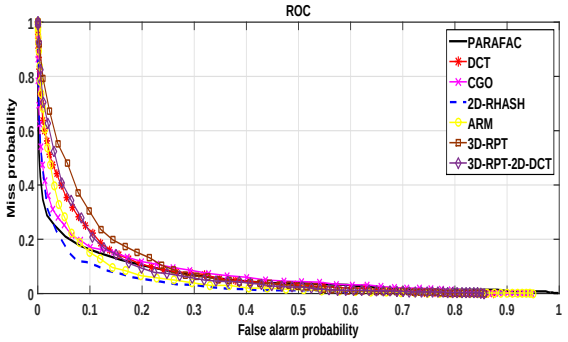
Figure 3.14: Statistical evaluation of video hashing algorithms via the ROC curves under reverse play attack.

normalized version of the original video and the performance of the algorithms are evaluated. This attack is repeated by inserting eight, sixteen and thirty-two unwanted frames, and then the performance of the algorithms is evaluated. The ROC curves under this attack are shown in Figure 3.15. This is a malicious attack, and hence the video hashing algorithm having the largest area under the ROC curve is said to be performing better. When 4 or 8 different video frames are inserted, all the video hashing algorithm has failed to capture the malicious attack because of being perceptually robust. As the number of unwanted frame insertion increases, the video hashing algorithm based on Scheme-I and Scheme-II of 3D-RASH perform better followed by the video hashing algorithms based on the PARAFAC decomposition, the ARM, the 2D-RASH, the CGO and the 3D-DCT.

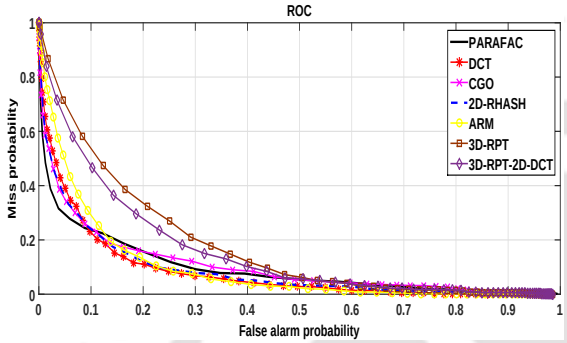
Watermark insertion attack: In this attack, a visible logo of different sizes are inserted to each video frame as a visible watermark. The following sizes of logo are considered: 16×16, 32×32, 48×48 and 64×64. The ROC curves for this attack are shown in Figure 3.16. This is also considered to be a malicious attack, and hence the video hashing algorithm having the largest area under the ROC curve is considered to be performing better. For the size of the logo up to 48×48 the performance of the video hashing algorithms based on the 2D-RASH, the CGO and the PARAFAC decomposition is having less area under the ROC curve while the video hashing algorithms based on the ARM, Scheme-I and Scheme-II of 3D-RASH and the



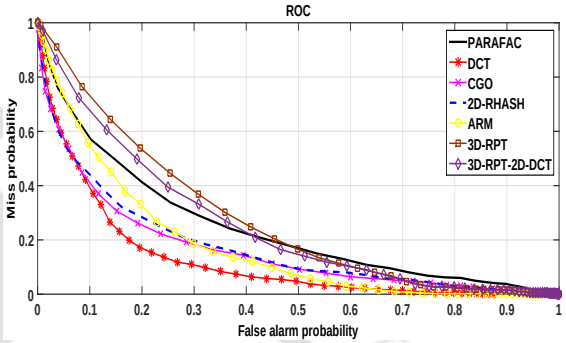
(a) 4 unwanted frames inserted.



(b) 8 unwanted frames inserted.

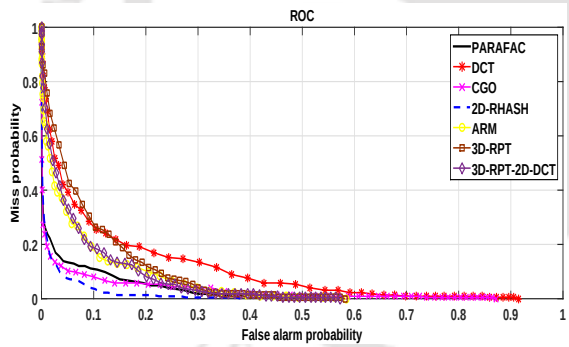


(c) 16 unwanted frames inserted.

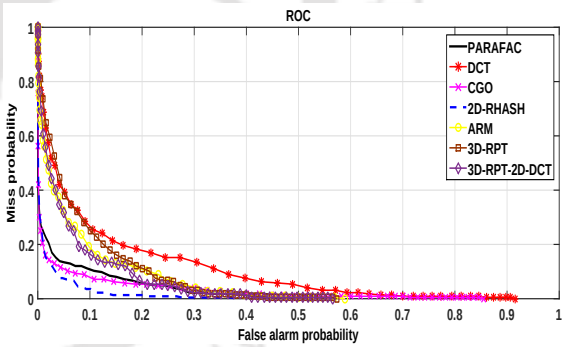


(d) 32 unwanted frames inserted.

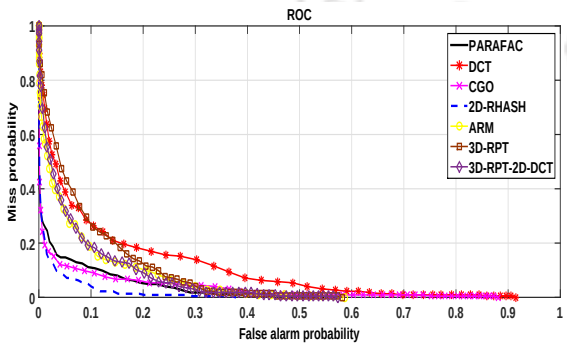
Figure 3.15: Statistical evaluation of video hashing algorithms via the ROC curves under frame insertion attack.



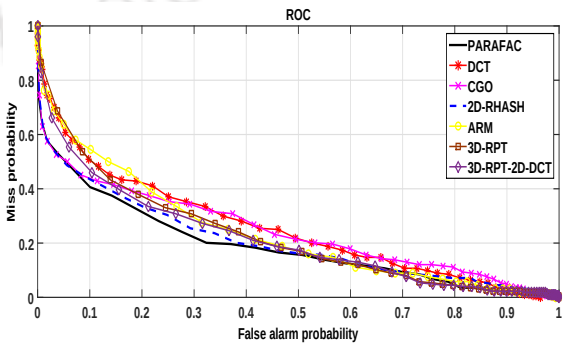
(a) Watermark of size 16×16.



(b) Watermark of size 32×32.

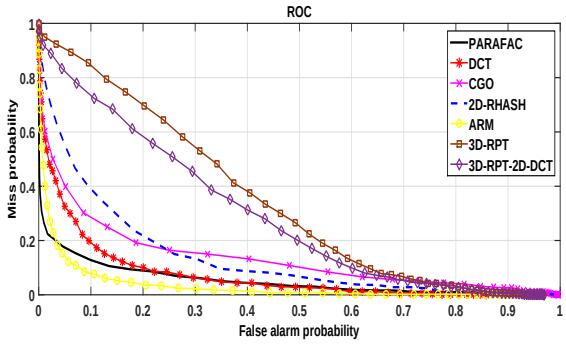


(c) Watermark of size 48×48.

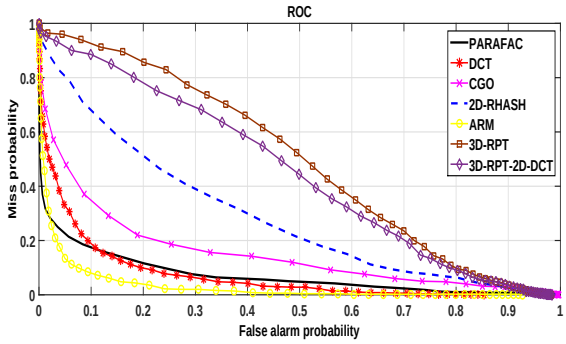


(d) Watermark of size 64×64.

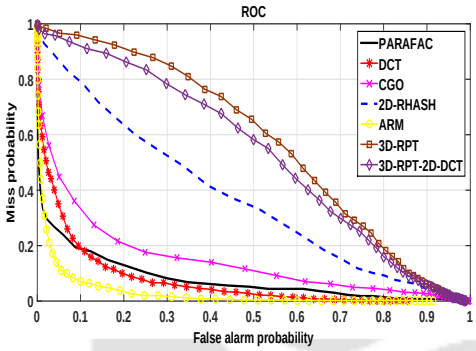
Figure 3.16: Statistical evaluation of video hashing algorithms via the ROC curves under watermark insertion attack.



(a) Average blurring using a 3×3 mask and adding zero mean AWGN with $\sigma = 81$.



(b) Average blurring using a 5×5 mask and adding zero mean AWGN with $\sigma = 114$.



(c) Average blurring using a 9×9 mask and adding zero mean AWGN with $\sigma = 127$.

Figure 3.17: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN and average blurring attacks.

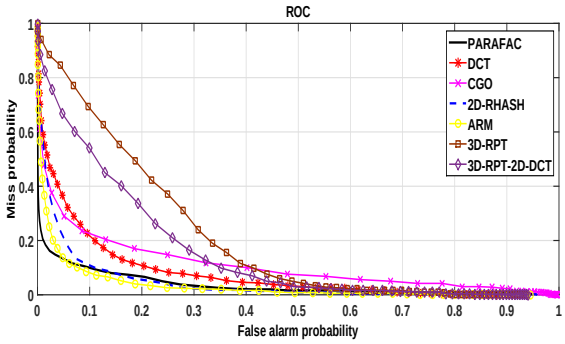
3D-DCT is have more area under the ROC curve. For the logo of size 64×64, the performance of all the video hashing algorithms considered is similarly good.

3.3.2.2 Multiple Attacks

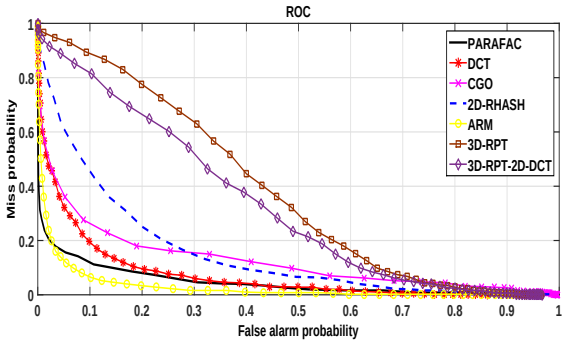
The evaluation of hash performance for various multiple attacks is as follows. The attacks include combination of various single image processing and malicious attacks described earlier.

AWGN and average blurring attacks: In this attack, each video frame was subjected to the average blurring with different mask sizes (3×3, 5×5 and 9×9), and then the addition of the zero mean AWGN with σ ranging from 81 to 127. The corresponding ROC curves are shown in Figure 3.17. The performance of the video hashing algorithms based on the 2D-RASH, Scheme-I and Scheme-II of the 3D-RASH degrades quickly as the value of σ increases. The performance of the video hashing algorithms based on the ARM, the PARAFAC decomposition, the 3D-DCT and the CGO is relatively stable.

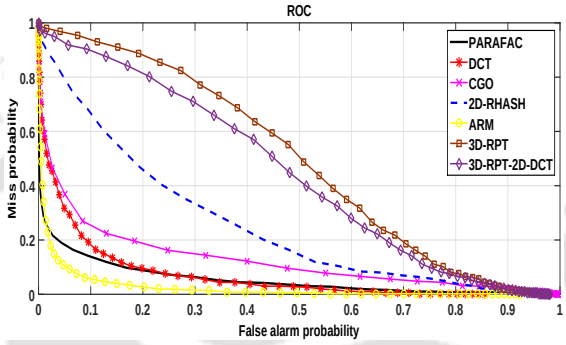
Salt and pepper noise and average blurring attacks: In this attack, each video frame was subjected to the average blurring with different mask sizes (3×3, 5×5 and 9×9), and then the addition of the salt and pepper noise with densities ranging from 5% to 25%. The corresponding ROC curves are shown in Figure 3.18. The performance of the video hashing algorithms based on the 2D-RASH, Scheme-I and Scheme-II of 3D-RASH degrades quickly as the percentage of noise density increases. The performance of the video hashing algorithms



(a) Average blurring using a 3×3 mask and adding a salt and pepper noise with 5% noise density.



(b) Average blurring using a 5×5 mask and adding a salt and pepper noise with 20% noise density.



(c) Average blurring using a 9×9 mask and adding a salt and pepper noise with 25% noise density.

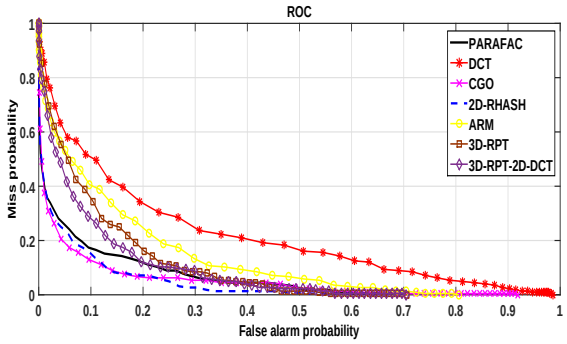
Figure 3.18: Statistical evaluation of video hashing algorithms via the ROC curves under salt and pepper noise and average blurring attacks.

based on the ARM, the PARAFAC decomposition, the 3D-DCT and the CGO is relatively stable.

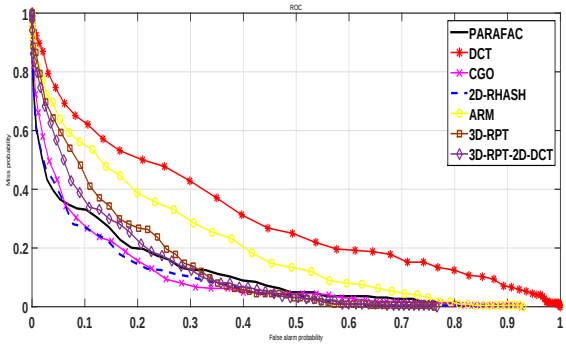
Cropping and Gaussian blurring attacks: In this attack, each video frame was subjected to the Gaussian filtering with different mask sizes (3×3, 5×5 and 9×9), and then the cropping of frame boundary pixels (5%, 10% and 15%). The corresponding ROC curves are shown in Figure 3.19. In all the cases, the performance of all the video hashing algorithms is good except for the ARM and the 3D-DCT based video hashing algorithms showing relatively poor performance.

AWGN and Gaussian blurring attacks: In this attack, each video frame was corrupted with zero mean AWGN with σ ranging from 57 to 114 and followed by the Gaussian filtering with different mask sizes (3×3, 5×5 and 9×9). The corresponding ROC curves are shown in Figure 3.20. It can be observed that in all the three cases, the performance of the video hashing algorithms based on the 2D-RASH, the PARAFAC decomposition, the 3D-DCT and the ARM is relatively stable. Further, the performance of the video hashing algorithms based on the CGO, Scheme-I and Scheme-II of the 3D-RASH is relatively unstable and weak.

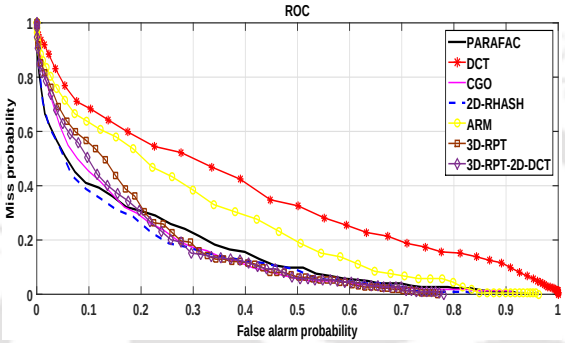
Frame-drop and interpolation followed by rotation attacks: In this attack video frames are dropped at regular intervals while retaining only 16 or 8 of the original frames and then interpolated to obtain 64 number of frames, followed by 5° or 8° rotation attacks respectively. The ROC curves for these multiple attacks are shown in Figure 3.21. In the first



(a) Gaussian blurring using a 3×3 mask and 5% cropping.



(b) Gaussian blurring using a 5 × 5 mask and 10% cropping.



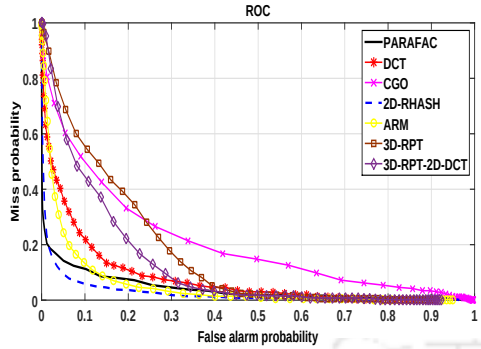
(c) Gaussian blurring using a 9 × 9 mask and 15% cropping.

Figure 3.19: Statistical evaluation of video hashing algorithms via the ROC curves under cropping and Gaussian blurring attacks.

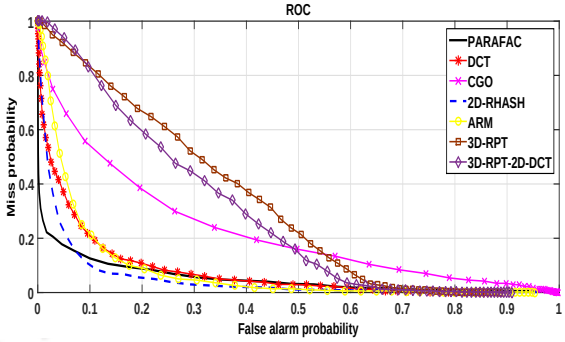
case of frame dropping and interpolation and the rotation attacks, the performance of all the video hashing algorithms is good except for the CGO based video hashing algorithm. In the second case of frame dropping and interpolation and the rotation attacks, the video hashing algorithm based on Scheme-I and Scheme-II of the 3D-RASH is relatively stable followed by the 2D-RASH and the ARM-based video hashing algorithms. The performance of the video hashing algorithms based on the PARAFAC decomposition, the 3D-DCT and the CGO is relatively weak.

Increased spatial resolution, frame-dropping and interpolation followed by rotation attacks: In this attack, the spatial resolution of each frame was increased to 500 × 500 or 1000 × 1000, subjected to frame dropping at regular intervals while retaining only 32 or 16 of the original frames and then interpolated to obtain 64 number of frames, followed by 5° or 8° rotation attacks respectively. The ROC curves for these multiple attacks are shown in Figure 3.22. In the first case, the performance of the video hashing algorithms based on the PARAFAC decomposition, the ARM and the 3D-DCT is relatively stable, and the remaining video hashing algorithms considered show relatively poor performance. In the second case, the performance of all the video hashing algorithms has deteriorated. The video hashing algorithm based on the CGO show poor performance in both the cases.

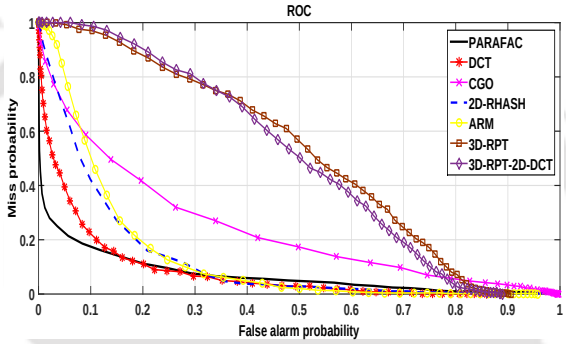
Decreased spatial resolution, frame-dropping and interpolation followed by average blurring attacks: In this attack, the spatial resolution of each frame was decreased



(a) Adding zero mean AWGN with $\sigma = 57$ and Gaussian blurring using a 3×3 mask.

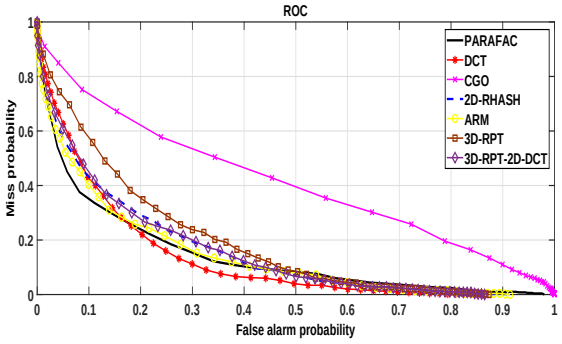


(b) Adding zero mean AWGN with $\sigma = 81$ and Gaussian blurring using a 5×5 mask.

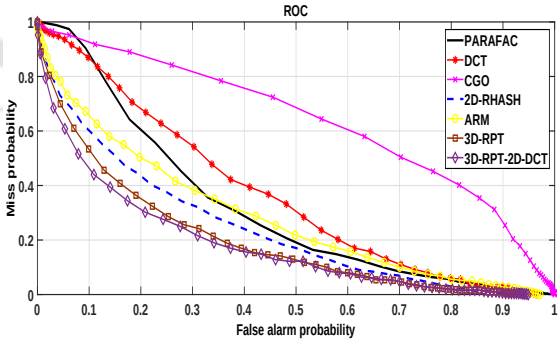


(c) Adding zero mean AWGN with $\sigma = 114$ and Gaussian blurring using a 9×9 mask.

Figure 3.20: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN and Gaussian blurring attacks.

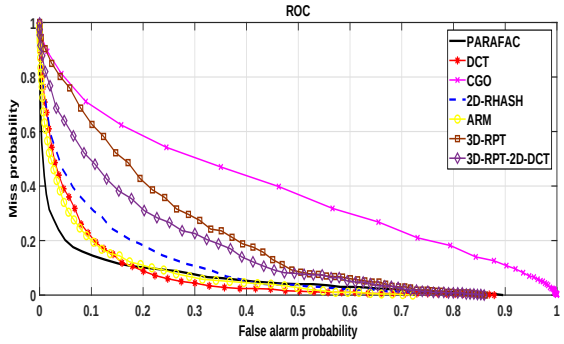


(a) Only 16 frames retained and 5° rotation.

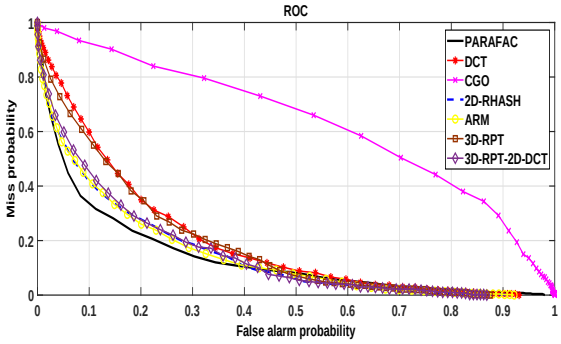


(b) Only 8 frames retained and 8° rotation.

Figure 3.21: Statistical evaluation of video hashing algorithms via the ROC curves under frame-drop and interpolation followed by rotation attacks.

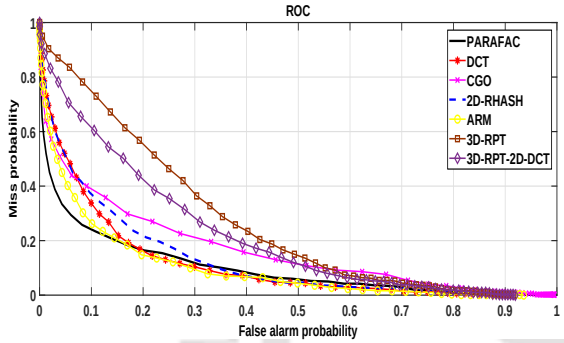


(a) Increase in spatial resolution to 500 × 500, only 32 frames retained and 5° rotation.

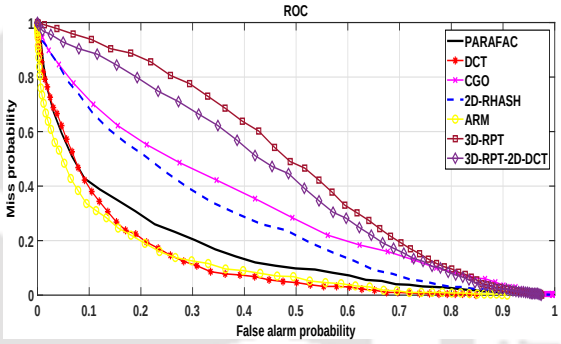


(b) Increase in spatial resolution to 1000 × 1000, only 16 frames retained and 8° rotation.

Figure 3.22: Statistical evaluation of video hashing algorithms via the ROC curves under increased spatial resolution, frame-dropping and interpolation followed by rotation attacks.



(a) Decrease in spatial resolution to 250 × 250, only 32 frames retained and average blurring using 5 × 5 mask.

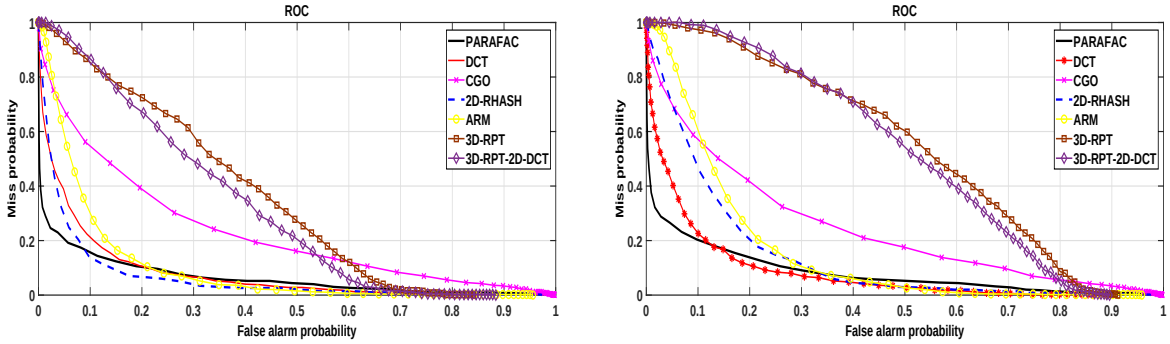


(b) Decrease in spatial resolution to 100 × 100, only 16 frames retained and average blurring using 9 × 9 mask.

Figure 3.23: Statistical evaluation of video hashing algorithms via the ROC curves under decreased spatial resolution, frame-dropping and interpolation followed by average blurring attacks.

to 250 × 250 or 100 × 100, subjected to frame dropping at regular intervals while retaining only 32 or 16 of the original frames and then interpolated to obtain 64 number of frames, followed by mean blurring using a mask of 5 × 5 or 9 × 9 respectively. The ROC curves for these multiple attacks are shown in Figure 3.23. In the first case, the performance of the video hashing algorithms based on the PARAFAC decomposition, the ARM, the 3D-DCT and 2D-RASH is similarly good. The other video hashing algorithms considered show relatively poor performance. In the second case, the performance of the video hashing algorithms based on the ARM, the PARAFAC decomposition and the 3D-DCT is relatively good while the other video hashing algorithms considered show relatively poor performance.

Addition of AWGN, frame insertion and Gaussian blurring attacks: In this attack, the video frames were subjected to zero mean AWGN with $\sigma = 81$ or $\sigma = 114$, insertion of 4 or 8 unwanted frames and the Gaussian filtering with a mask size of 5 × 5 or 9 × 9 respectively. The corresponding ROC curves are shown in Figure 3.24. This is a combination of content-preserving attack and the malicious attack. The video hashing algorithms based on the PARAFAC decomposition, the 3D-DCT, the 2D-RASH and the ARM are having a relatively



(a) Adding a zero mean AWGN with $\sigma = 81$, 4 unwanted frames inserted and Gaussian blurring using a 5×5 mask.

(b) Adding a zero mean AWGN with $\sigma = 114$, 8 unwanted frames inserted and Gaussian blurring using a 9×9 mask.

Figure 3.24: Statistical evaluation of video hashing algorithms via the ROC curves under addition of AWGN, frame insertion and Gaussian blurring attacks.

smaller area under the ROC curves and fail to capture the malicious attack. While the video hashing algorithms based on the CGO, Scheme-I and Scheme-II of the 3D-RASH having a relatively larger area under the ROC curves and hence capable of capturing the malicious attack.

3.4 Concluding Remarks on ARM-based Perceptual Video Hash

The performance of the proposed video hashing algorithm was found to be robust to the following single image processing attacks: average blurring, Gaussian blurring, addition of AWGN, salt and pepper noise, rotation, cropping, modified brightness, modified contrast, frame dropping and interpolation, modified spatial resolutions, compression, and insertion of a logo as the watermark. For most of the multiple attacks, the performance of the proposed video hashing algorithm was found to be comparable to the other video hashing algorithms considered.

From the validation and the performance evaluation of the hash, it can be concluded that the performance of the ARM-based video hashing algorithm is better than the 3D-RASH based video hashing algorithm for most of the typical image processing attacks but still has the scope for improvement when compared to the video hashing algorithms considered. Furthermore, the ARM based video hashing algorithm is computationally simple compared to the 3D-RASH based video hashing algorithm.

3.5 Perceptual Video Hashing using TWT and ARM

In [28], Mani et al., have generated the video hash based on temporally informative representative images (TIRIs). The TIRI was generated using weighted superposition of luminance values along the temporal direction of a predefined number of frames chosen from a given video. The image hashes are generated from the individual TIRIs, and then concatenated to generate the video hash. The resulting hashes were robust to a range of attacks including noise addition,

rotation, time shift, and frame dropping. In [45], Saikia and Bora, have computed the video hash at a group-of-frames level from the spatiotemporal low-pass bands of the wavelet transformed groups-of-frames. Their work demonstrated the robustness of the video hash function against the content-preserving and content changing attacks. The prior works in [45] and [28] motivate us to generate the video hash in two phases. In the first phase, the dimensionality in the temporal direction is reduced by applying the TWT and considering only the low pass frames for hashing. In the second phase, the dimensionality is further reduced by applying a random projection based technique on the low pass frames to generate a compact hash that satisfies the perceptual hashing properties discussed in Section 1 of Chapter 1.

Thus, extending the work in Section 3.2, instead of applying the random projection technique directly on the overlapping normalized sub-videos, here, the temporal dimension is reduced by applying the TWT in the temporal direction and considering the low pass frames for hashing. It is observed that the application of the TWT along the temporal direction naturally generates the temporally informative representative frames (TIRFs) [46]. Later, these TIRFs were subjected to random projections. The proposed technique has the following advantages:

1. The TWT applied on the luminance values along the temporal direction of the video frames captures the important visual features of the sequence in the low pass frames. Considering only the low pass frames reduces the dimensions and hence the computational complexity in the further steps.

2. The projections of points from high dimensional data to low dimensional data through a random projection technique approximately preserve the length of the unit vectors and also the pairwise distance among the points.

3. The high computational efficiency and security due to the random projection make it suitable for practical implementation.

Thus, the combined TWT and the random projection technique reduces the dimensionality and at the same time, extracts the essential features present in the input video data.

3.6 Temporal Wavelet Transform

It is quite known that the level of information within a video changes from frame to frame gradually or abruptly. Some groups of frames contain significantly more information requiring detailed analysis, while the other groups of frames may contain less information requiring coarse analysis. Under such circumstances, the multiresolution representation of the video by applying the DWT facilitates a simple and efficient hierarchical framework for interpreting the information in the video [47]. The DWT uses the principle of subband coding for the multiresolution (different scales and shifts) representation of discrete signals. This representation provides useful signal information localized in time, space and frequency to a greater extent. In subband coding, the signal is passed through a series of low-pass and high-pass filters to subdivide the signal into several frequency bands. Figure 3.25 shows the basic procedure of the DWT decomposition of the signal [48]. The coefficients of the low-pass filter are given by $(0.7071, 0.7071)$

and the coefficients of high pass filter is given by $(-0.7071, 0.7071)$ corresponding to the Haar wavelet. Most of the important information of the input signal is present at the output of the coarse approximation filter.

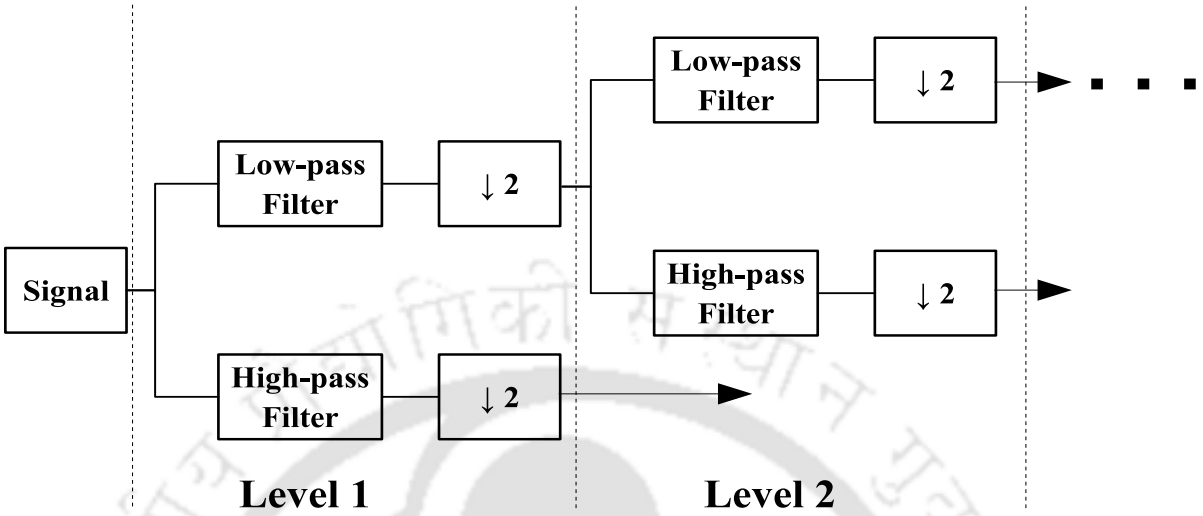


Figure 3.25: 1D DWT using the low-pass and high-pass filter.

The combination of such low pass and high pass filters are called a quadrature mirror filter pair. The orthogonal wavelets such as Haar wavelets, Daubechies wavelets and Coiflets satisfy the quadrature mirror filter relationship. The number of times a signal passes through a low pass/high pass filter indicates the level of decomposition.

Applying the TWT on the input video data results in a subband of approximation frames (temporal low-frequency components) and subbands of detail frames (temporal high-frequency components). Since the Haar wavelet is relatively simple and faster compared to the other wavelets, it is used as the mother wavelet for multiresolution representation of the video along the temporal direction to produce the static and the dynamic components of the video.

The number of frames in the video must be equal to the power of two. If the number of video frames is not equal to the power of two, then the additional frames obtained through the temporal interpolation of the input video frames are inserted to make it equal to the power of two.

3.7 Proposed Video Hashing Algorithm based on TWT and ARM

The generic block diagram of the proposed video hashing algorithm based on the TWT and random projection is shown in Figure 3.26. The algorithm consists of the following blocks: *preprocessing and normalization*, *TWT*, *randomization*, *random projections* and *hash computation*. The algorithm generates the hash by capturing the spatiotemporal essence of the input video and hence, belongs to the class of spatiotemporal video hashing algorithms. The brief description of the blocks is as follows:

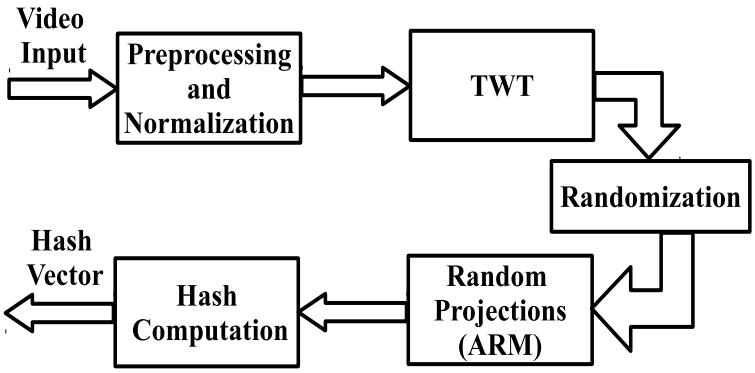


Figure 3.26: The block diagram representation for the extraction of perceptual video hash using the TWT and the ARM.

(i) *Preprocessing and Normalization*: Since the resultant hash size of the algorithm is kept constant, we standardize the spatial size of the videos to $X \times X$ via spatial re-sizing. Further, the video is transformed from the RGB colour space to the grey level. This normalization process ensures that the effect of various spatial dimensions and chrominance components on the hash value to be minimal. Let the normalized video be represented by $\mathcal{V}_{norm}$.

(ii) *TWT*: The TWT requires the number of frames in $\mathcal{V}_{norm}$ to be a power of two. If the number of frames is not equal to a power of two then, the temporal interpolation is done to insert frames in between the original frames to make it equal to the nearest power of two. Next, apply the TWT on $\mathcal{V}_{norm}$ in the temporal direction using a suitable wavelet to obtain the transformed video. Decide the number of levels of decomposition, depending on the number of frames in the video. Then, select a predefined number of low-pass frames obtained at the last level of decomposition. Let $\mathcal{V}_{norm-trans}$ be this normalized and transformed video. For example, if the number of frames in the video is 1000 then, using temporal interpolation operation insert frames in between the original frames and make it equal to 1024 frames. Apply TWT up to four levels of decomposition on this video in the temporal direction to obtain the transform domain frames. Select 64 low-pass frames obtained after the fourth level of decomposition for further processing. 64 frames were chosen in the proposed video hashing algorithm to have a fair performance comparison with the existing video hashing algorithms, as all the algorithms retain 64 frames to generate the hash.

The remaining steps of this algorithm to generate a compact perceptual video hash is the same as that of the ARM-based perceptual video hashing algorithm proposed in Section 3.2. The algorithmic steps of the TWT-ARM based hashing method are summarized in Algorithm 3.2.

Algorithm 3.2 *Proposed video hashing algorithm based on the TWT and the ARM.*

1: **Input**:

- RGB video V_{in} and k (even) ;
- Algorithm parameters: X, U, K, N, ϵ and β;

2: **Preprocessing and Normalization**:

- Convert V_{in} into a sequence of grey-level frames, V_{grey}
- Normalize V_{grey} via spatial re-sizing to obtain $\mathcal{V}_{norm}$ of spatial size $X \times X$

3: **TWT**:

- If $remainder\left(\log_2\left(No.of Frames\right), 1\right)! = 0$ then, increase the number of frames by temporal averaging so that $remainder\left(\log_2\left(No.of Frames\right), 1\right) == 0$
- Apply TWT in temporal direction up to l^{th} level of decomposition
- Retain X number of low pass frames to get the normalized transform video $\mathcal{V}_{norm-trans}$

4: **Randomization**:

- Depending on the secret key K, randomly select N overlapping 3D sub-cubes from $\mathcal{V}_{norm-trans}$, to form $\mathcal{V}_i$, $i = 1, 2, ..., N$, each of size $U \times U \times U$, such that $U < X$
- Concatenate the columns of each sub-cube $\mathcal{V}_i$ to form vector $\mathbf{f}_i \in \mathbb{R}^d$ such that $d = U \times U \times U$
- Construct the feature matrix $\mathbf{F} \in \mathbb{R}^{d \times N}$ from $\mathbf{f}_i$ s to obtain $\mathbf{F} = [\mathbf{f}_1\, \mathbf{f}_2 \ldots \mathbf{f}_N]$

5: **Random projections**: Apply the dimensionality reduction technique to $\mathbf{F}$ using the ARM projection matrix $\mathbf{R}_{ARM} \in \mathbb{R}^{k \times d}$ to get the hash matrix $\mathbf{H} = \frac{1}{\sqrt{k}}\mathbf{R}_{ARM}\mathbf{F}$

6: **Hash computation**:

- Generate the intermediate hash vector $\mathbf{h}' = \frac{\sum_{i=1}^{N} \mathbf{h}(:,i)}{N}$, where $\mathbf{h}(:, i)$ is the i^{th} column of the matrix $\mathbf{H}$.
- Rank order $\mathbf{h}'$ to obtain $\mathbf{h}'' = [h'_{(1)}\, h'_{(2)} \ldots h'_{(k)}]^T$.
- Median, $\xi = \left(\frac{h'\left(\frac{k}{2}\right) + h'\left(\frac{k}{2}\right)+1}{2}\right)$.
- Binarize the elements of $\mathbf{h}'$ to obtain $\mathbf{h} = [h_1\, h_2 \ldots h_k]^T$ where

$$h_j = \begin{cases} 0; & h'_j < \xi \\ 1; & h'_j \geq \xi \end{cases},\ j = 1, 2, \ldots, k.$$

7: **Output**

- A hash vector $\mathbf{h}$

3.8 Simulation Results and Discussion

1000 test sequences are taken from [33], [34], [35] and [36] are considered to evaluate the performance of the video hashing algorithms. The details of the database are tabulated in Table 2.1 and discussed in Chapter 2. A number of experiments are conducted to validate the hash properties and evaluate the performance of the hash.

All frames were spatially normalized to 64×64. After applying the TWT, the 64 low-pass frames were retained. 32 randomly overlapping 3D sub-cubes of size $16 \times 16 \times 16$ were chosen from the $64 \times 64 \times 64$ normalized video, using a secret key such that it covers the entire video. The number 32 was decided by trial and error. Thus, each feature vector is of size 4096. To have a fair comparison with other algorithms, a uniform hash length of 128 is considered. Accordingly, the values of ϵ and β in Achlioptas's theorem were chosen to be 0.725 and 0.5, respectively.

3.8.1 Hash Validation

Experiments were conducted to validate the unpredictability, the visual fragility and the perceptual robustness properties of the hash generated using the TWT-ARM based video hashing algorithm. 224 videos from the database and 1000 secret keys are considered to validate the proposed hash. The NHD was the distance metric used to compare the hash values of the videos. The histograms of the NHD to evaluate the important desirable properties of the hash are shown in Figure 3.27. The validations are described below.

(i) *To validate the unpredictability property*
128-bit hashes were generated for a test sequence using 1000 different secret keys, and then, the NHD was measured between every possible pair of the hashes. The procedure was repeated for the remaining 223 test sequences. The histogram of the NHD values for this experiment is shown in Figure 3.27(a). The figure shows that the NHD measured between the hashes in this case approximately follows a Gaussian distribution with the mean 0.5 and standard deviation of 0.089. A narrow-spread distribution centred around 0.5 implies that the perceptual hash closely satisfies the unpredictability property given in Equation 1.5. Comparing Figure 3.27(a) and Figure 3.2(a), we can conclude that the perceptual hash based on the TWT-ARM satisfies the unpredictability property better than the perceptual hash based on the ARM.

(ii) *To validate the visual fragility property*
128-bit hashes were generated for 224 test sequences using the same secret key, and then, the NHD was measured between every possible pair of the hashes. The procedure was repeated for the remaining 999 secret keys. The histogram of the NHD values for this experiment is shown in Figure 3.27(b). The figure shows that the NHD measured between the hashes in this case approximately follows a Gaussian distribution with the mean centred around 0.5 and standard deviation of 0.176. A narrow-spread distribution centred around 0.5 implies that the perceptual hash closely satisfies the visually fragile property given

Table 3.1: Comparison of hash validity among the ARM and the TWT-ARM based perceptual video hashing algorithms.

Name of the Video Hashing Algorithm	Unpredictability		Visual Fragility		Perceptual Robustness	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
ARM	**0.5**	**0.106**	**0.5**	**0.193**	**0.25**	**0.147**
TWT ARM	**0.5**	**0.089**	**0.5**	**0.176**	**0.225**	**0.132**

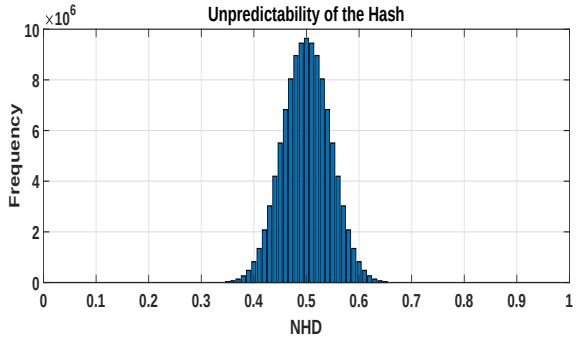
in Equation 1.4. Comparing Figure 3.27(b) and Figure 3.2(b), we can conclude that the perceptual hash based on the TWT-ARM satisfies the visual fragility property better than the perceptual hash based on the ARM.

(iii) *To validate the perceptual robustness property*
All the videos existing in the database were corrupted with AWGN of $\sigma = 81$ and subsequently blurred with the Gaussian mask of size 5×5 thereby generating a set of perceptually similar videos. The hashes were generated for these videos using the same secret key. The NHD between the hashes of the original videos and the perceptually similar videos was measured. A histogram skewed towards zero implies that the perceptual hash closely satisfies the perceptual robustness property given in Equation 1.3. The histogram of all the NHD values for this experiment is shown in Figure 3.27(c) and we observe that the NHD between the original videos and the perceptually similar videos are distributed between 0 and approximately 0.45 with the mean centred around 0.225. Comparing Figure 3.27(c) and Figure 3.2(c), we conclude that the perceptual hash based on the TWT-ARM satisfies the perceptual robustness property better than the perceptual hash based on the ARM.

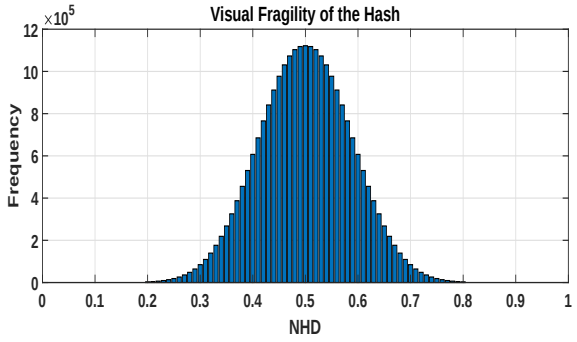
(iv) *To assess perceptual robustness versus visual fragility*
To evaluate the results of perceptual similarity obtained from the proposed video hashing algorithm for the perceptually similar and perceptually distinct videos, the histograms of the NHD obtained for the perceptual robustness property and the visual fragility property are superimposed as shown in Figure 3.27(d). From the figure, it can be observed that the histogram of the NHD for the perceptually similar and distinct videos overlap. The overlapping indicates that there would be misclassification between the perceptually similar and different videos. This figure also shows that the overlap of these two superimposed histograms is quite less implying lower misclassification rates. Further, the overlapping here is less compared to Figure 3.2(d). Hence, the performance of the video hashing algorithm based on the TWT-ARM is better than the video hashing algorithm based on the ARM.

The unpredictability, visual fragility and the perceptual robustness properties of the perceptual hashes generated based on the TWT-ARM and the ARM are compared in terms of the mean and the standard deviation of the NHD in Table 3.1.

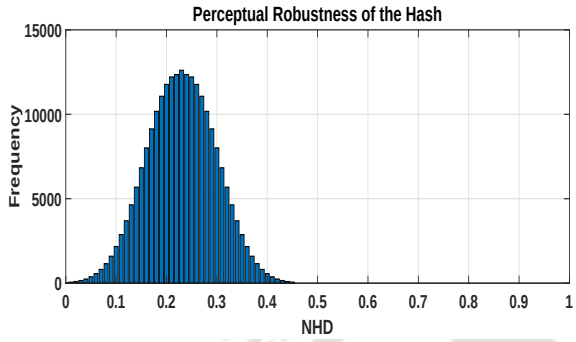
From Figure 3.27, Figure 3.2 and Table 3.1 we conclude that the perceptual hash generated using the TWT-ARM satisfies all the desirable properties more efficiently.



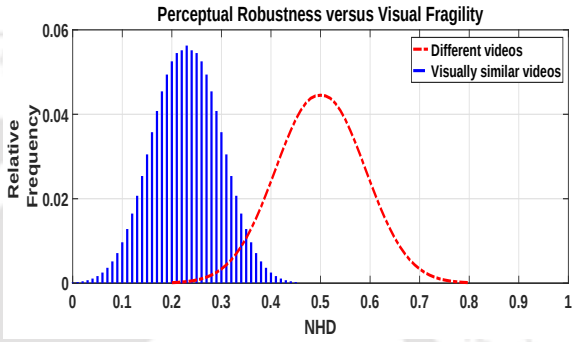
(a) Unpredictability property of the hash.



(b) Visual fragility property of the hash.



(c) Perceptual robustness property of the hash.



(d) Perceptual robustness property versus the visual fragility of the hash.

Figure 3.27: The histogram of normalized Hamming distance to evaluate the important desirable properties of the hash generated using the TWT-ARM based video hashing algorithm.

3.8.2 Evaluation of Hash Performance

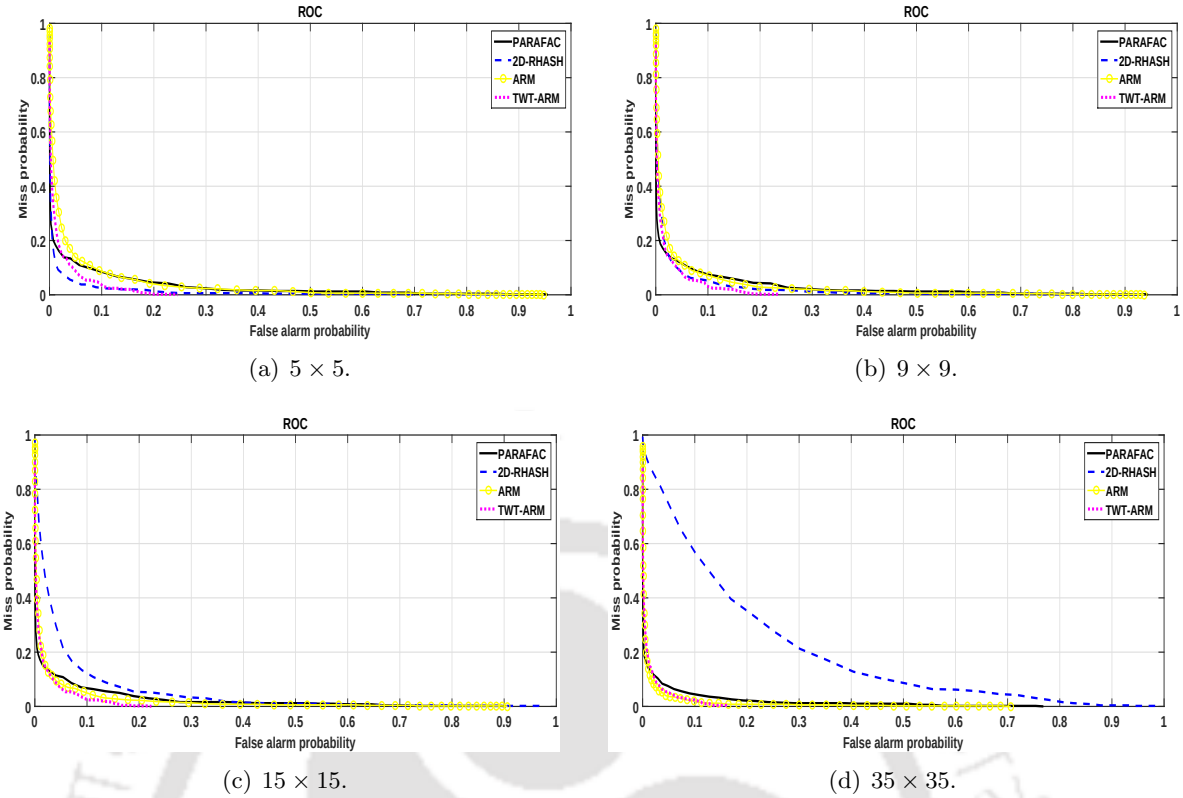
The ROC curve is used to evaluate the performance of the algorithm. The same single/multiple image processing attacks as in Chapter 2 are considered here. In this set of experiments, the performance of the TWT-ARM based video hashing algorithm is compared with the following perceptual video hashing algorithms.

- the PARAFAC decomposition based video hashing algorithm [15], labelled as PARAFAC in the ROC curve.
- the 2D radial projection based video hashing algorithm [11, 10], labelled as 2D-RHASH in the ROC curve.
- the ARM-based video hashing algorithm, labelled as ARM in the ROC curve.

These algorithms showed competitive performance against most of the common image processing attacks and hence were chosen for comparison. The TWT-ARM based video hashing algorithm is labelled as TWT-ARM in the ROC curve.

3.8.2.1 Single Attacks

The evaluation of hash performance for various single attacks is as follows. The attacks include the image processing and the malicious attacks described earlier.



(a) 5×5. (b) 9×9.

(c) 15×15. (d) 35×35.

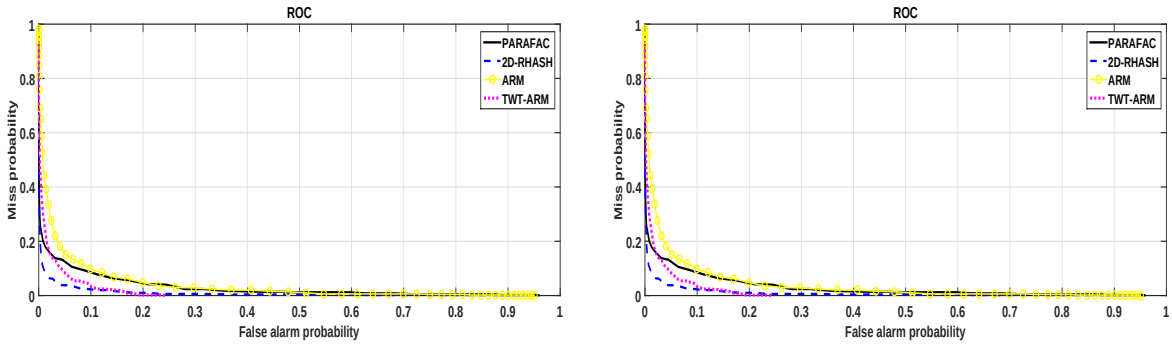
Figure 3.28: Statistical evaluation of video hashing algorithms via the ROC curves under average blurring attack using different mask size.

Average blurring attack: Figure 3.28 shows the ROC curves under average blurring using the mask of the following sizes: 5×5, 9×9, 15×15 and 35×35. The performance of the video hashing algorithms based on the PARAFAC decomposition, the ARM and the TWT-ARM is similarly good for the mean filter of all mask sizes considered. The performance of the video hashing algorithm based on the 2D-RASH is relatively poor for the mean filter of mask sizes 15×15 and 35×35.

Gaussian blurring attack: The ROC curves under the Gaussian filtering using the 5×5, 9×9, 15×15 and 35×35 masks are shown in Figure 3.29. The performance of all the video hashing algorithms under considerations is similarly good for all the mask sizes. It can be observed that the performance of all the video hashing algorithms considered deteriorates as the mask size is increased to 15×15 and 35×35.

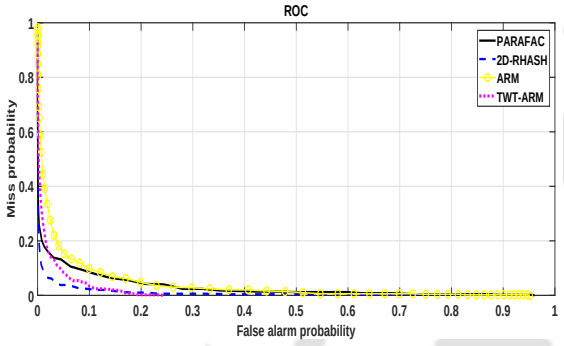
AWGN attack: Figure 3.30 shows the ROC curves for the zero mean AWGN attack with the following values of σ: 57, 81, 114 and 127. The ARM-based video hashing algorithm is relatively stable followed by the PARAFAC decomposition and the TWT-ARM based video hashing algorithms. As the value of σ increases, the performance of the video hashing algorithm based on the 2D-RASH deteriorates quickly.

Salt and pepper noise attack: Figure 3.31 shows the ROC curves under the salt and pepper noise attack with the following noise densities: 5%, 10%, 20% and 25%. The ARM-based video hashing algorithm is relatively stable followed by the PARAFAC decomposition and the TWT-ARM based video hashing algorithms. It can be observed that as the density of the noise

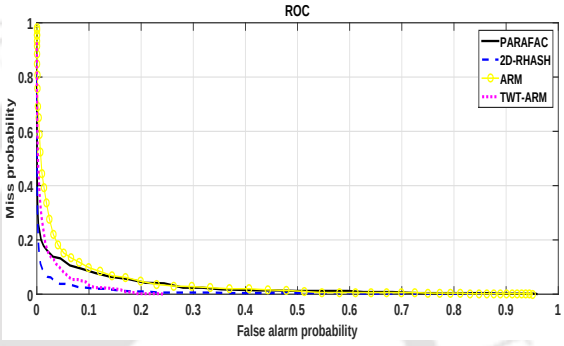


(a) 5×5.

(b) 9×9.

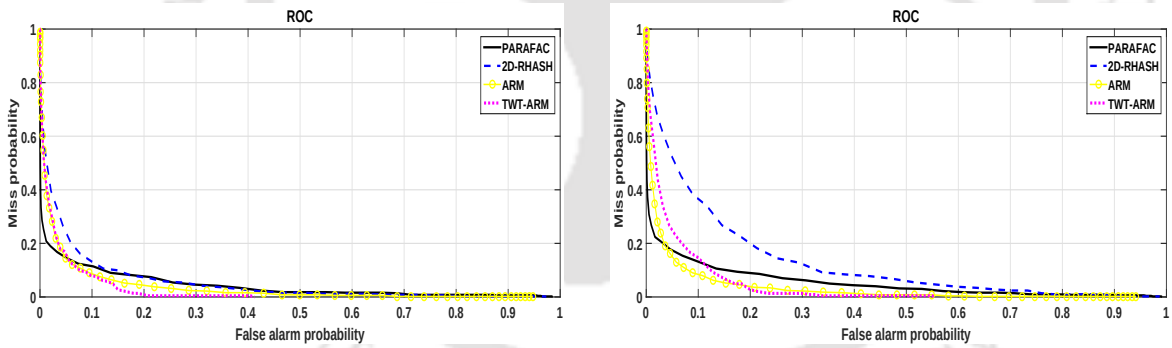


(c) 15×15.



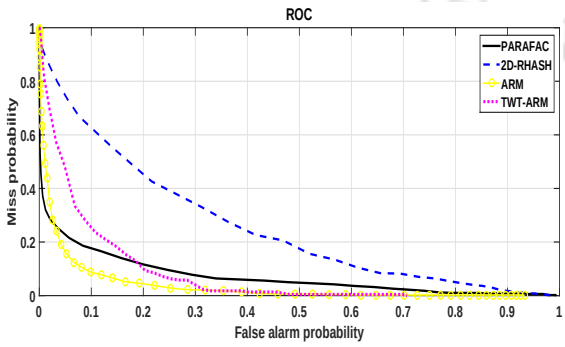
(d) 35×35.

Figure 3.29: Statistical evaluation of video hashing algorithms via the ROC curves under Gaussian blurring attack using different mask size.

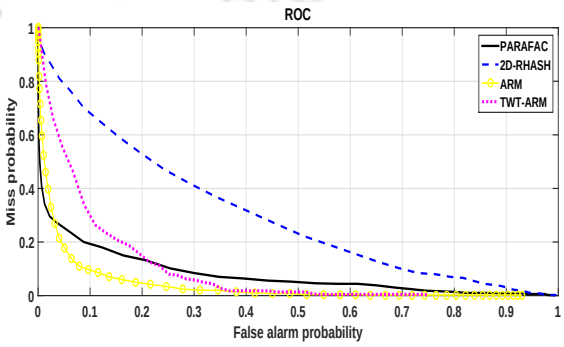


(a) Zero mean AWGN with $\sigma = 57$.

(b) Zero mean AWGN with $\sigma = 81$.

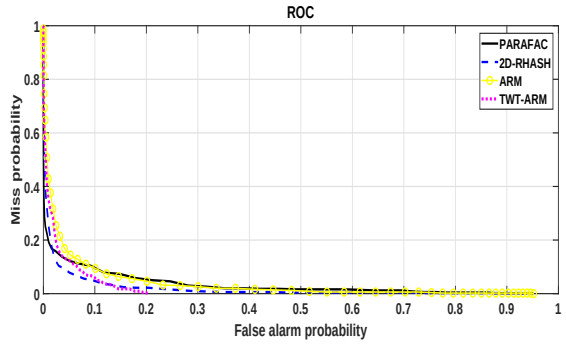


(c) Zero mean AWGN with $\sigma = 114$.

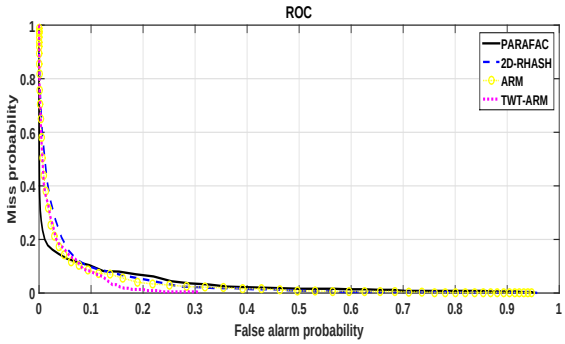


(d) Zero mean AWGN with $\sigma = 127$.

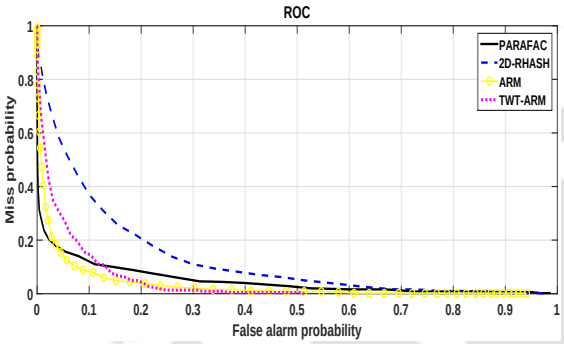
Figure 3.30: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN attack.



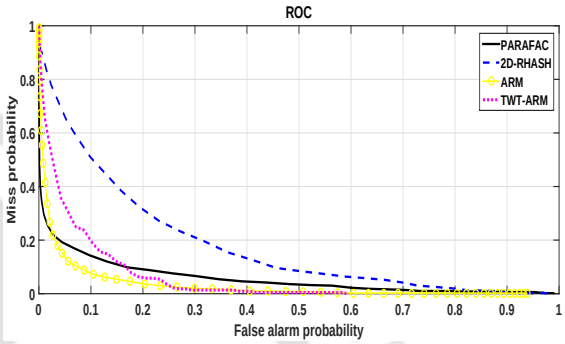
(a) Salt and pepper noise with 5% noise density.



(b) Salt and pepper noise with 10% noise density.



(c) Salt and pepper noise with 20% noise density.



(d) Salt and pepper noise with 25% noise density.

Figure 3.31: Statistical evaluation of video hashing algorithms via the ROC curves under salt and pepper noise attack.

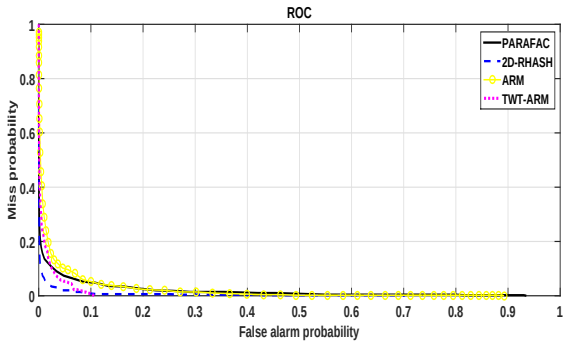
increases, the performance of the video hashing algorithm based on the 2D-RASH degrades quickly.

Rotation attack: Figure 3.32 shows the ROC curves under the rotation attack with the following degrees of rotation: 2°, 5°, 8° and 12°. The performance of all the video hashing algorithms considered is nearly the same for 2°, 5° and 8° rotation attack. For 12° rotation attack, the ARM-based video hashing algorithm has deteriorated.

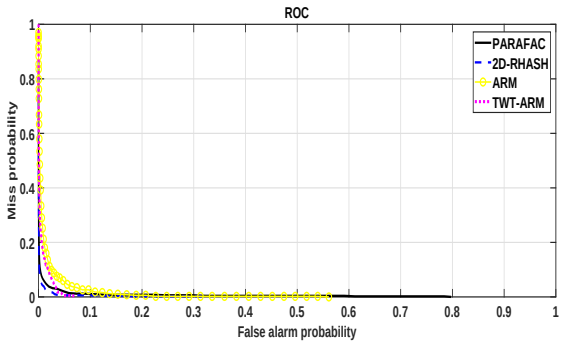
Cropping attack: The ROC curves for the cropping attack are shown in Figure 3.33. The following percentages of pixels were cropped from the borders of each frame: 5%, 10% and 15%. The performance of all the video hashing algorithms considered is good and relatively the same except the ARM-based video hashing algorithm. It can be observed from the figure that as more and more boundary pixels are cropped, the performance of the video hashing algorithm based on the ARM becomes progressively poor compared to the other video hashing algorithms.

Brightness variation attack: The ROC curves under the modified brightness attack (±5% and ±10%) are shown in Figure 3.34. The performance of the 2D-RASH based video hashing algorithm is relatively stable. The performance of all the other video hashing algorithms considered is more or less the same except that of the TWT-ARM based video hashing algorithm showing relatively poor performance.

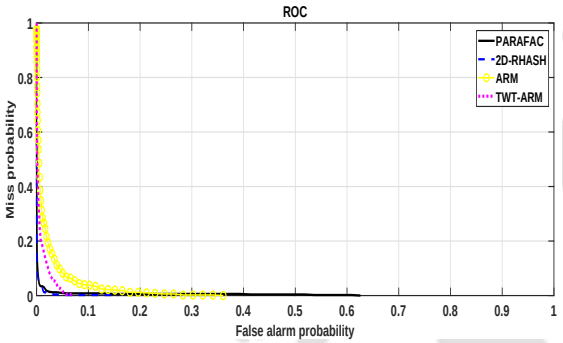
Contrast variations attack: The ROC curves for the modifications in the contrast are shown in Figure 3.35. The contrast of the frames are increased by multiplying the frame pixels with 1.1 and 1.2, and decreased by multiplying the frame pixels with 0.9 and 0.8, respectively.



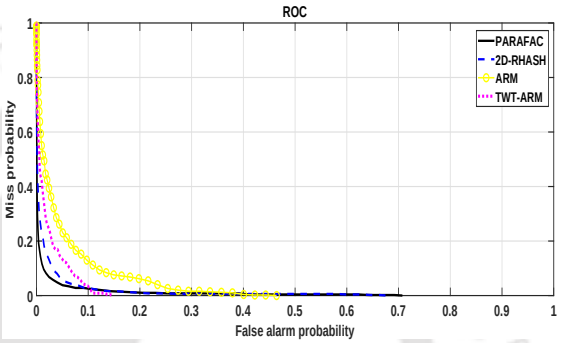
(a) 2°.



(b) 5°.

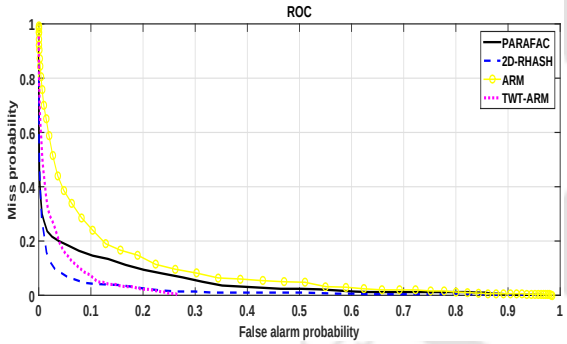


(c) 8°.

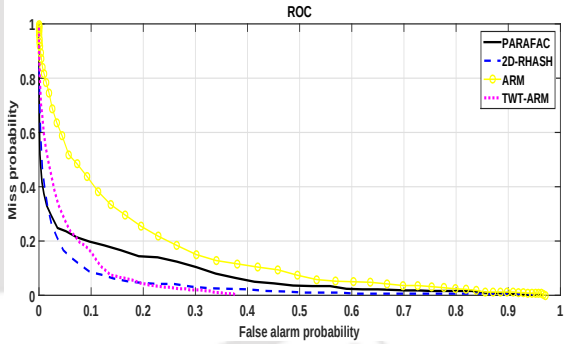


(d) 12°.

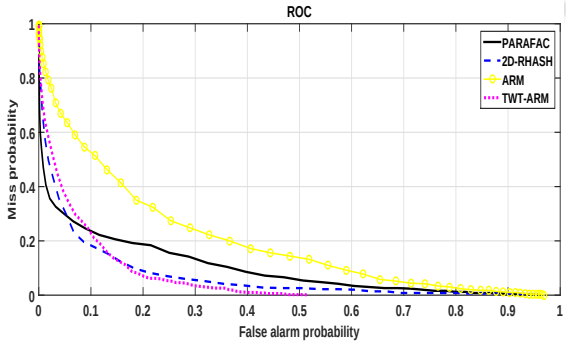
Figure 3.32: Statistical evaluation of video hashing algorithms via the ROC curves under rotation attack with different degrees of frame rotation.



(a) 5% crop.

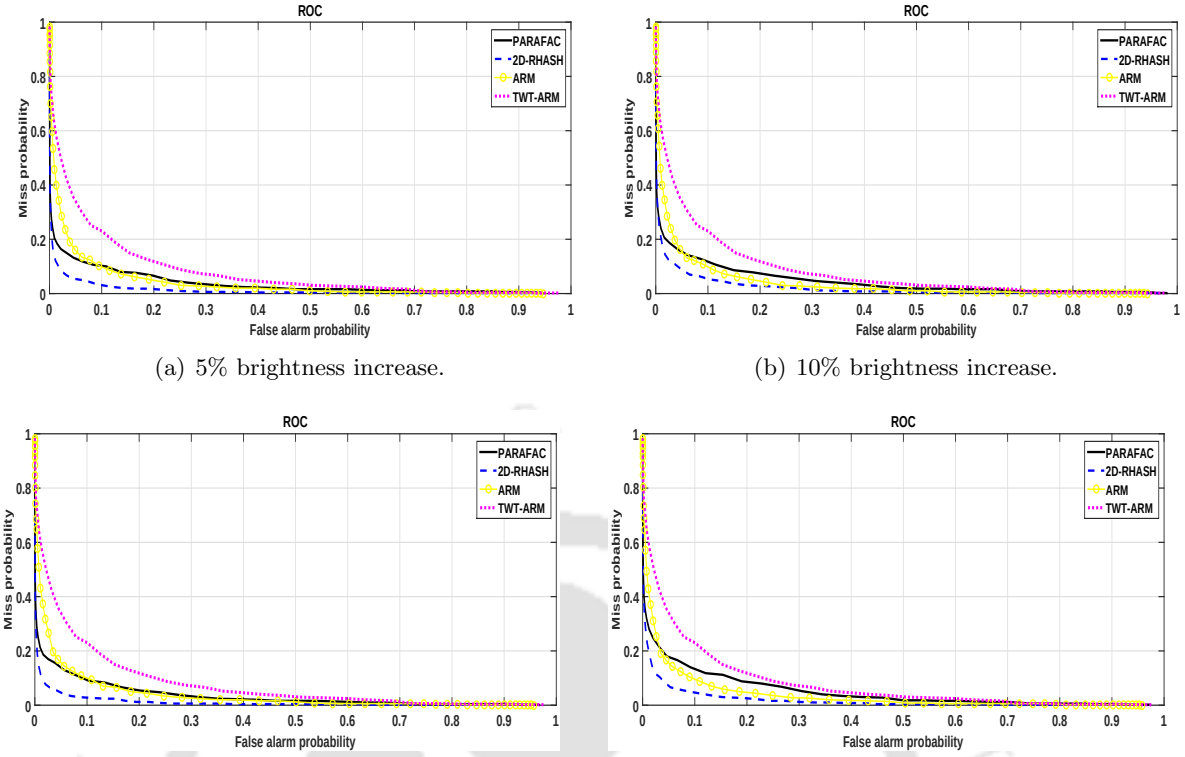


(b) 10% crop.



(c) 15% crop.

Figure 3.33: Statistical evaluation of video hashing algorithms via the ROC curves under cropping attack with different percentages of crop.



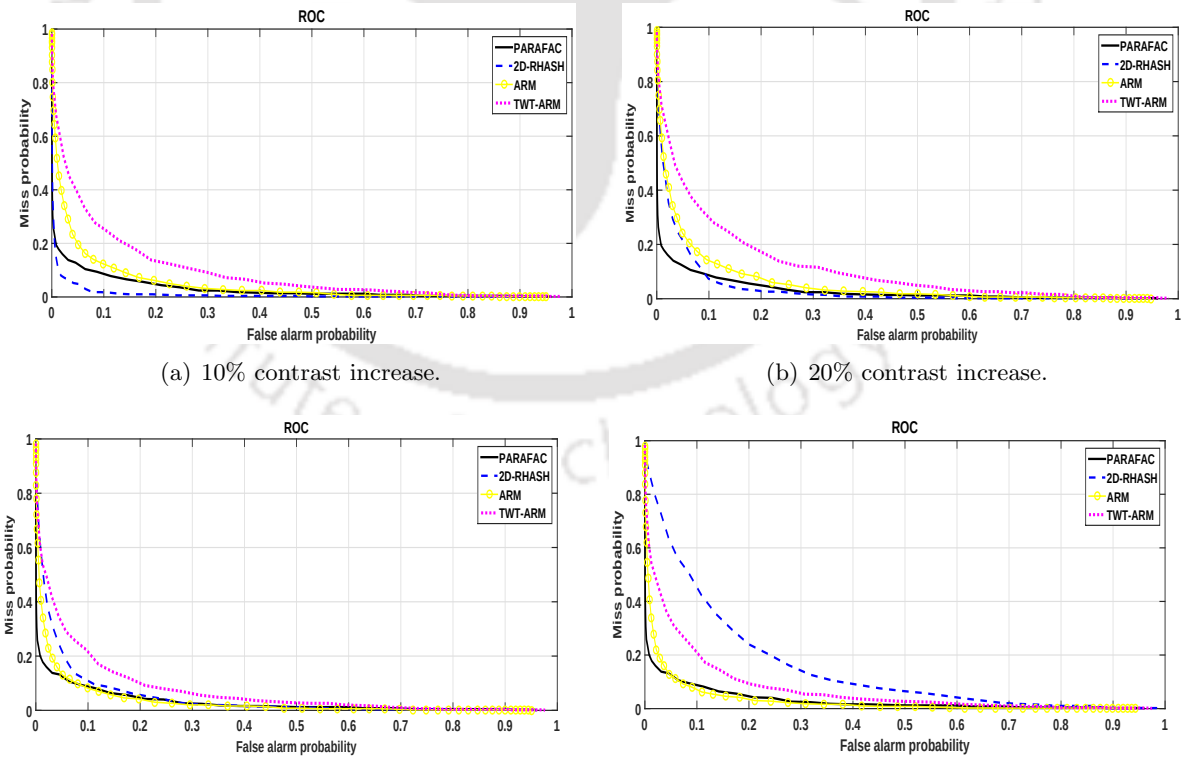
(a) 5% brightness increase.

(b) 10% brightness increase.

(c) 5% brightness decrease.

(d) 10% brightness decrease.

Figure 3.34: Statistical evaluation of video hashing algorithms via the ROC curves under brightness variation attack.



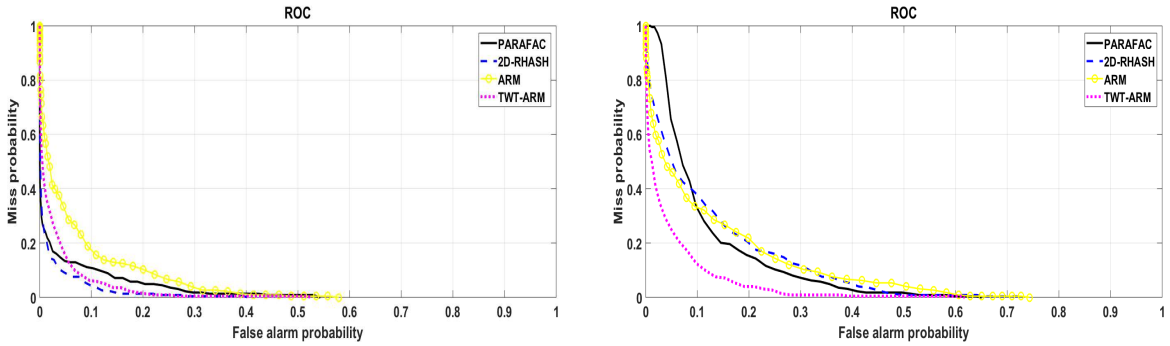
(a) 10% contrast increase.

(b) 20% contrast increase.

(c) 10% contrast decrease.

(d) 20% contrast decrease.

Figure 3.35: Statistical evaluation of video hashing algorithms via the ROC curves under contrast variation attack.



(a) 48 frames retained.

(b) 32 frames retained.

Figure 3.36: Statistical evaluation of video hashing algorithms via the ROC curves under frame-dropping and interpolation attack.

The video hashing algorithm based on the PARAFAC decomposition show robustness to contrast variations compared to the other video hashing algorithms considered. The TWT-ARM based video hashing algorithm is also relatively stable, but the area under the ROC curve is more. The 2D-RASH based video hashing algorithm is least robust to the contrast decreased attack.

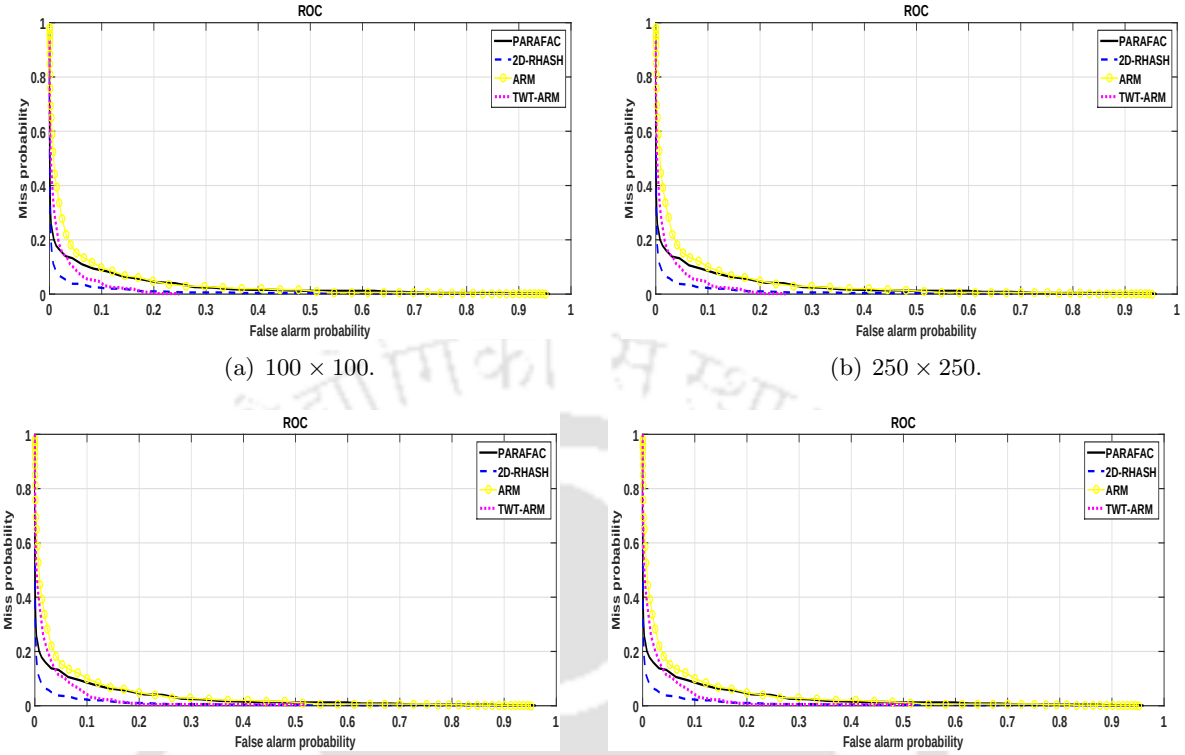
Frame-dropping and interpolation attack: The ROC curves for the frame-dropping and interpolation attack are shown in Figure 3.36. In the first case, only 48 frames were retained, and then the remaining 16 frames were interpolated using temporal averaging. It can be observed that all the video hashing algorithms considered show similar performance except the ARM-based video hashing showing relatively poor performance. In the second case, only 32 frames were retained, and then the remaining 32 frames were interpolated using temporal averaging. In this case, the TWT-ARM based video hashing algorithm showed the maximum robustness to this attack. The remaining algorithms show relatively poor performance.

Spatial resolution variation attack: Figure 3.37 shows the ROC curves under the modified spatial resolutions attack. The following spatial resolutions were considered: 100×100, 250×250, 500×500 and 1000×1000. It can be observed that all the algorithms under considerations are having almost the same area under the ROC curves implying that the performance of all the video hashing algorithms considered is similar.

Compression attack: Two video database was constructed with the average CR of 250:1 and 240:1, respectively. The statistical evaluation of video hashing algorithms using the ROC curves under the compression attack is shown in Figure 3.38. It can be observed that for both the CRs, the performance of the video hashing algorithm based on the 2D-RASH has the least area under the ROC curves followed by the TWT-ARM, the PARAFAC decomposition and the ARM-based video hashing algorithms.

Reverse play attack: The ROC curves under the reverse play attack are shown in Figure 3.39. The TWT-ARM based video hashing algorithm was successful in having the least false alarm probability and the miss probability. The ROC curves for the remaining video hashing algorithms stayed close to the line of no discrimination implying the poor performance of all the algorithms considered.

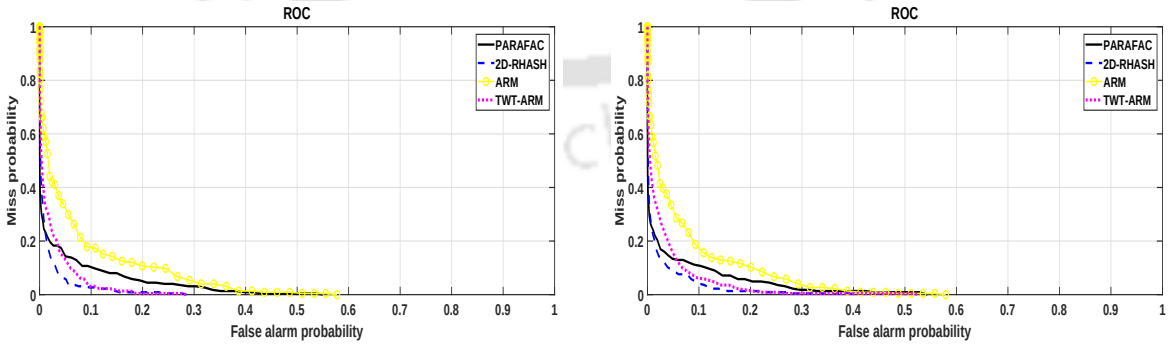
Frame insertion attack: Firstly, four frames of a different video are inserted in the



(a) 100×100. (b) 250×250.

(c) 500×500. (d) 1000×1000.

Figure 3.37: Statistical evaluation of video hashing algorithms via the ROC curves under spatial resolution variation attack.



(a) 100kbps bit-rate and 240:1 average CR. (b) 64kbps bit-rate and 250:1 average CR.

Figure 3.38: Statistical evaluation of video hashing algorithms via the ROC curves under compression attack.

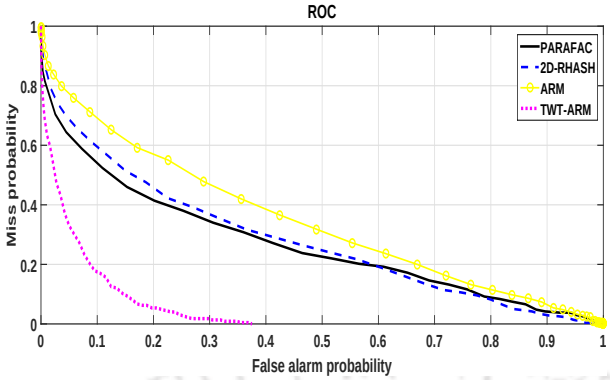
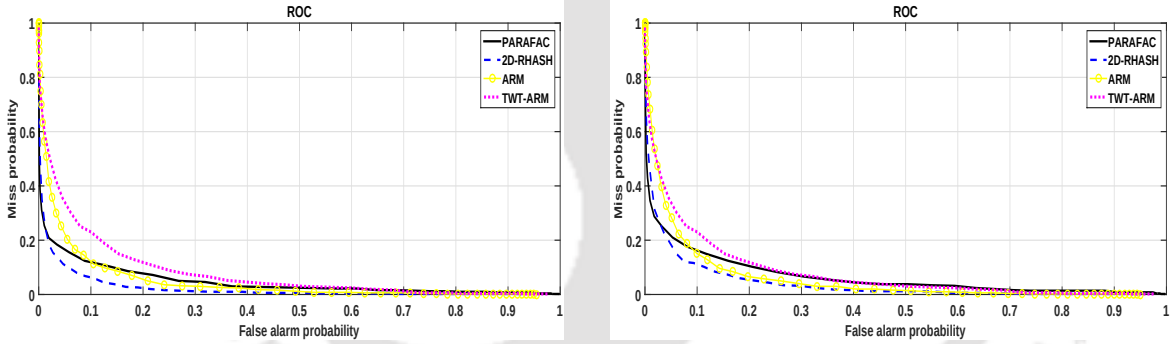
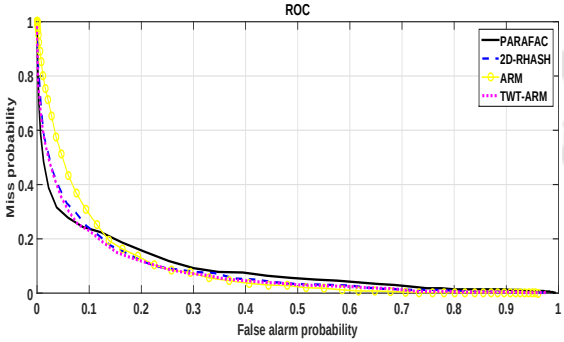


Figure 3.39: Statistical evaluation of video hashing algorithms via the ROC curves under reverse play attack.

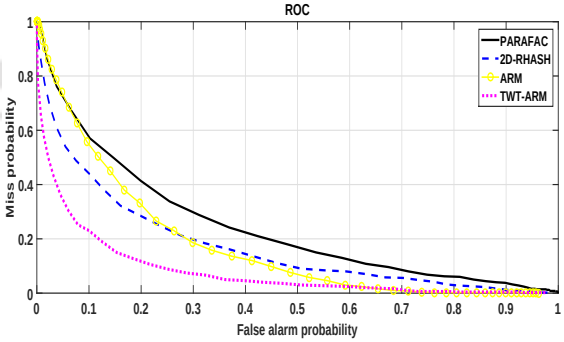


(a) 4 unwanted frames inserted.

(b) 8 unwanted frames inserted.

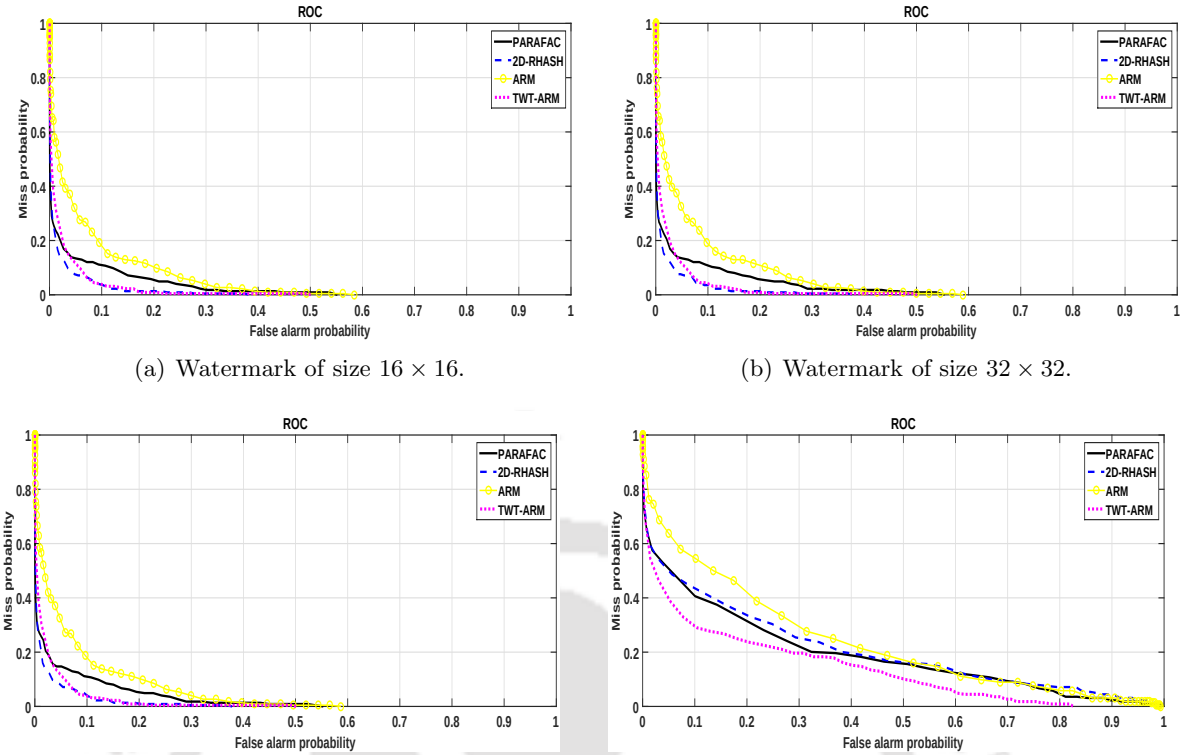


(c) 16 unwanted frames inserted.



(d) 32 unwanted frames inserted.

Figure 3.40: Statistical evaluation of video hashing algorithms via the ROC curves under frame insertion attack.



(a) Watermark of size 16×16.

(b) Watermark of size 32×32.

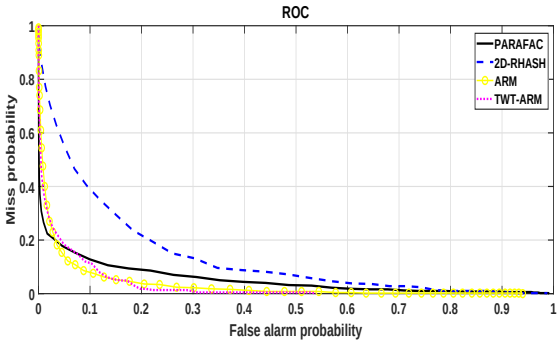
(c) Watermark of size 48×48.

(d) Watermark of size 64×64.

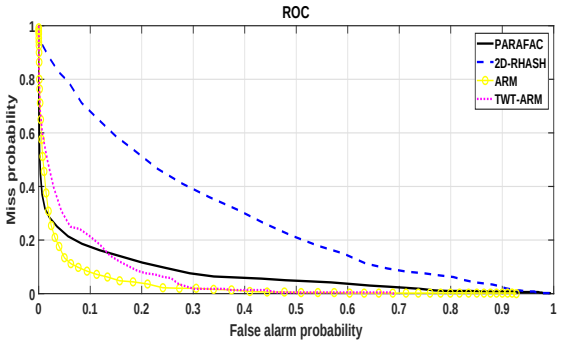
Figure 3.41: Statistical evaluation of video hashing algorithms via the ROC curves under watermark insertion attack.

normalized version of the original video, and the performance of the algorithms are evaluated. This attack is repeated by inserting eight, sixteen and thirty-two unwanted frames, and then the performance of the algorithms are evaluated. The ROC curves under this attack are shown in Figure 3.40. This is a malicious attack, and hence the video hashing algorithm having the largest area under the ROC curve is said to be performing better. It can be observed that the TWT-ARM based video hashing algorithm is relatively stable in all the cases. When 4 or 8 different video frames are inserted, all the video hashing algorithm has failed to capture the malicious attack because of being perceptually robust. As the number of unwanted frame insertion increases, the PARAFAC decomposition based video hashing algorithm perform better followed by the video hashing algorithms based on the ARM and the 2D-RASH.

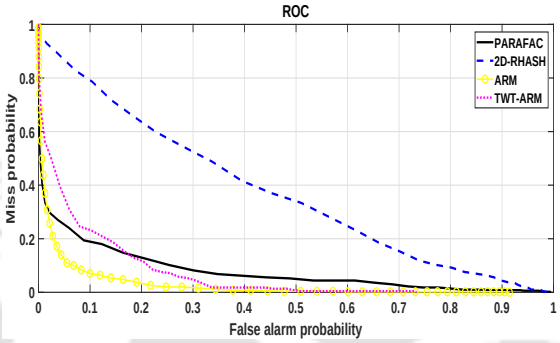
Watermark insertion attack: In this attack, a visible logo of different sizes is inserted to each video frame as a visible watermark. The following sizes of logo are considered: 16×16, 32×32, 48×48 and 64×64. The ROC curves for this attack are shown in Figure 3.41. This is also considered to be a malicious attack, and hence the video hashing algorithm having the largest area under the ROC curve is considered to be performing better. For the size of the logo up to 48×48 the performance of the video hashing algorithm based on the 2D-RASH is having less area under the ROC curve followed by the TWT-ARM, the PARAFAC decomposition, and the ARM-based video hashing algorithms having relatively more area under the ROC curve. For the logo of size 64×64, the performance of all the video hashing algorithms considered is similarly good.



(a) Average blurring using a 3×3 mask and adding zero mean AWGN with $\sigma = 81$.



(b) Average blurring using a 5×5 mask and adding zero mean AWGN with $\sigma = 114$.



(c) Average blurring using a 9×9 mask and adding zero mean AWGN with $\sigma = 127$.

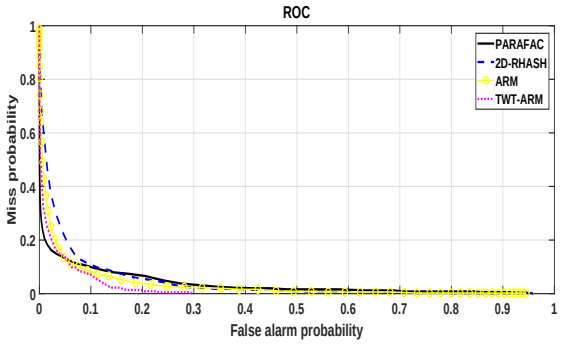
Figure 3.42: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN and average blurring attacks.

3.8.2.2 Multiple Attacks

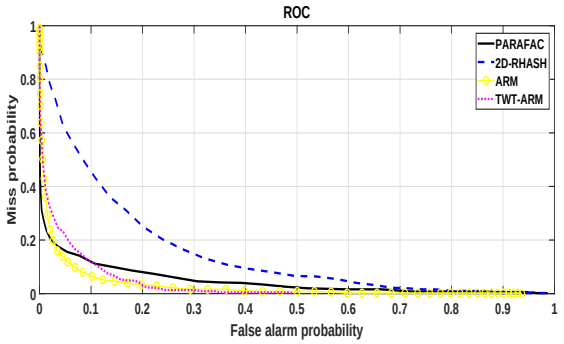
The evaluation of hash performance for various multiple attacks is as follows. The attacks include combination of various single image processing and malicious attacks described earlier.

AWGN and average blurring attacks: In this attack, each video frame was subjected to the average blurring with different mask sizes (3×3, 5×5 and 9×9), and then the addition of the zero mean AWGN with σ ranging from 81 to 127. The corresponding ROC curves are shown in Figure 3.42. For average blurring using a 3×3 mask and addition of zero mean AWGN with $\sigma = 81$, the performance of all the video hashing algorithms is good except for the 2D-RASH based video hashing algorithm. Similar observations are made when the video is subjected to average blurring using a 5×5 mask and zero mean AWGN with $\sigma = 114$ attacks. When the video is subjected to average blurring using a 9×9 mask and addition of zero mean AWGN with $\sigma = 127$, the video hashing algorithms based on the TWT-ARM and the PARAFAC decomposition have similar performance. The performance of the 2D-RASH based video hashing algorithm was poor under this attack in all the cases while the ARM-based video hashing algorithm was relatively stable.

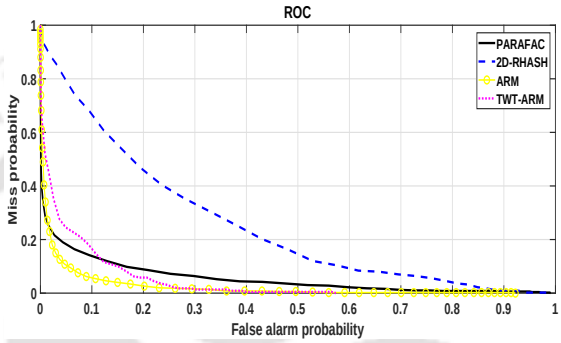
Salt and pepper noise and average blurring attacks: In this attack, each video frame was subjected to the average blurring with different mask sizes (3×3, 5×5 and 9×9), and then the addition of the salt and pepper noise with densities ranging from 5% to 25%. The corresponding ROC curves are shown in Figure 3.43. For average blurring using a 3×3 mask



(a) Average blurring using a 3×3 mask and adding a salt and pepper noise with 5% noise density.



(b) Average blurring using a 5×5 mask and adding a salt and pepper noise with 20% noise density.



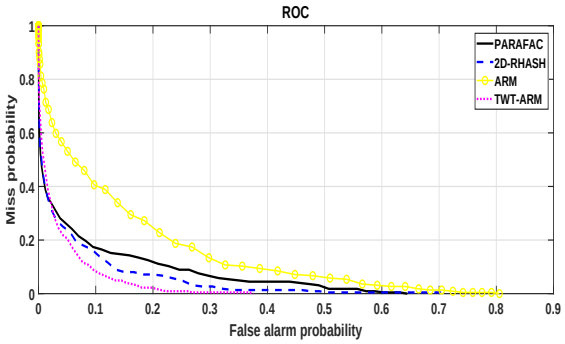
(c) Average blurring using a 9×9 mask and adding a salt and pepper noise with 25% noise density.

Figure 3.43: Statistical evaluation of video hashing algorithms via the ROC curves under salt and pepper noise and average blurring attacks.

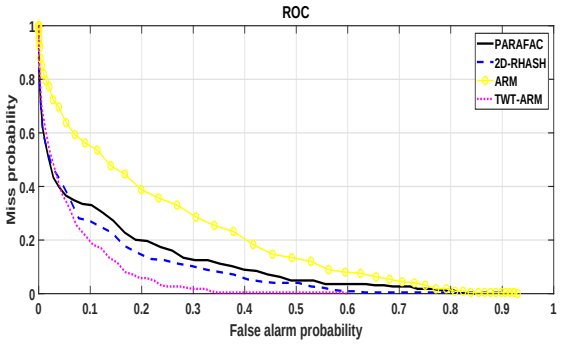
and addition of salt and pepper noise with 5% noise density, the performance of all the video hashing algorithms is good. For average blurring using a 5×5 mask and addition of salt and pepper noise with 20% noise density, the performance of all the video hashing algorithms is good except for the 2D-RASH based video hashing algorithm. Similar observations are made when the video is subjected to average blurring using a 9×9 mask and salt and pepper noise with 25% noise density attacks.

Cropping and Gaussian blurring attacks: In this attack, each video frame was subjected to the Gaussian filtering with different mask sizes (3×3, 5×5 and 9×9), and then the cropping of frame boundary pixels (5%, 10% and 15%). The corresponding ROC curves are shown in Figure 3.44. For the Gaussian blurring using a 3×3 mask and 5% cropping of frame boundary pixels, the performance of all the video hashing algorithms is good except for the ARM-based video hashing algorithm showing relatively poor performance. For the Gaussian blurring using a 5×5 mask and 10% cropping of frame boundary pixels, the performance of the TWT-ARM based video hashing algorithm is relatively good followed by the PARAFAC decomposition and the 2D-RASH based video hashing algorithms showing similar performance. The ARM-based video hashing algorithm has poor performance in this case. A similar observation is made for the Gaussian blurring using a 9×9 mask and 15% cropping of frame boundary pixels attacks.

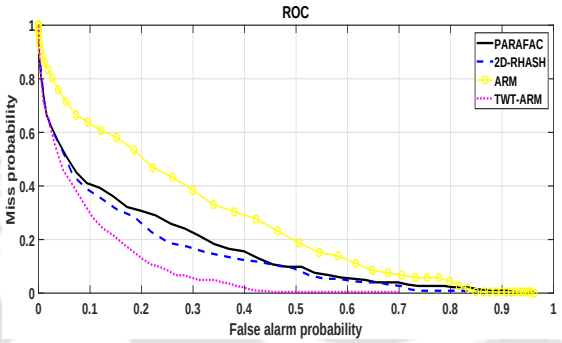
AWGN and Gaussian blurring attacks: In this attack, each video frame was corrupted



(a) Gaussian blurring using a 3×3 mask and 5% cropping.



(b) Gaussian blurring using a 5×5 mask and 10% cropping.



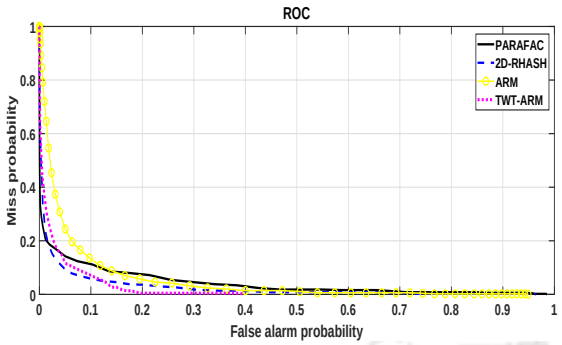
(c) Gaussian blurring using a 9×9 mask and 15% cropping.

Figure 3.44: Statistical evaluation of video hashing algorithms via the ROC curves under cropping and Gaussian blurring attacks.

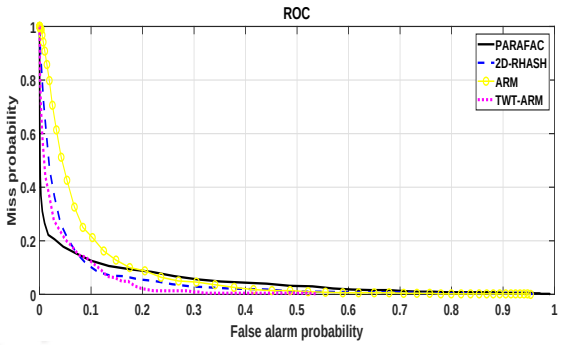
with zero mean AWGN with σ ranging from 57 to 114 and followed by the Gaussian filtering with different mask sizes (3×3, 5×5 and 9×9). The corresponding ROC curves are shown in Figure 3.45. For the addition of zero mean AWGN with $\sigma = 57$ and the Gaussian blurring using a 3×3 mask, the performance of all the video hashing algorithms is good. For the addition of zero mean AWGN with $\sigma = 81$ and the Gaussian blurring using a 5×5 mask, the performance of all the video hashing algorithms is good except for the ARM-based video hashing algorithm showing relatively poor performance. Finally, when the video is subjected to the addition of zero mean AWGN with $\sigma = 114$ and the Gaussian blurring using a 9×9 mask, the performance of all the video hashing algorithms is good except for the ARM and the 2D-RASH based video hashing algorithms showing relatively poor performance.

Frame-dropping and interpolation followed by rotation attacks: In this attack video frames are dropped at regular intervals while retaining only 16 or 8 of the original frames and then interpolated to obtain 64 number of frames, followed by 5° or 8° rotation attacks respectively. The ROC curves for these multiple attacks are shown in Figure 3.46. It can be observed that the video hashing algorithm based on the TWT-ARM has the least area under the ROC curves indicating better performance. While the remaining video hashing algorithms considered, show relatively poor performance.

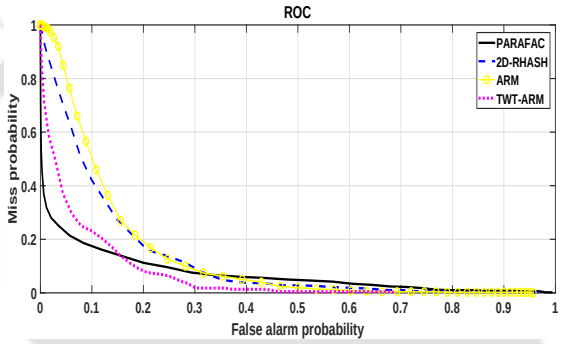
Increased spatial resolution, frame-dropping and interpolation followed by rotation attacks: In this attack, the spatial resolution of each frame was increased to 500×500



(a) Adding zero mean AWGN with $\sigma = 57$ and Gaussian blurring using a 3×3 mask.

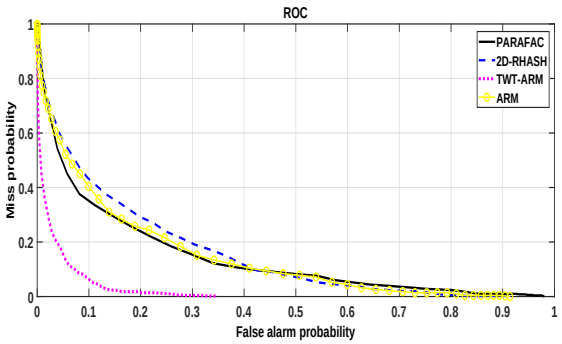


(b) Adding zero mean AWGN with $\sigma = 81$ and Gaussian blurring using a 5×5 mask.

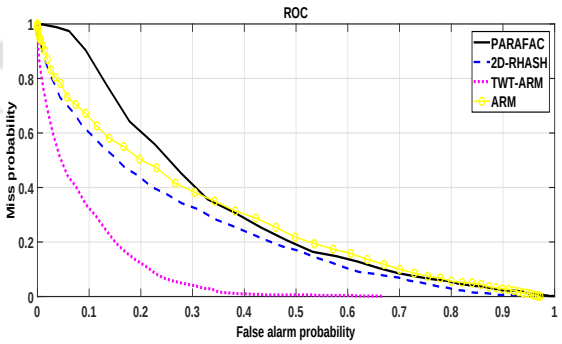


(c) Adding zero mean AWGN with $\sigma = 114$ and Gaussian blurring using a 9×9 mask.

Figure 3.45: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN and Gaussian blurring attacks.

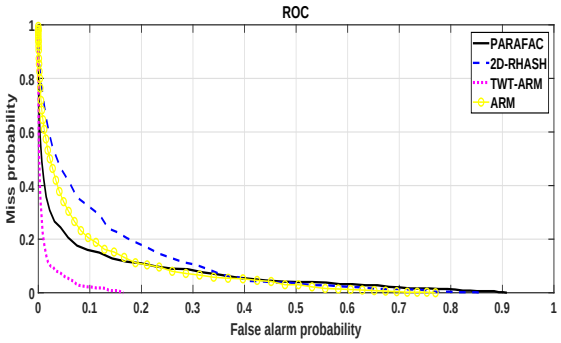


(a) Only 16 frames retained and 5° rotation.

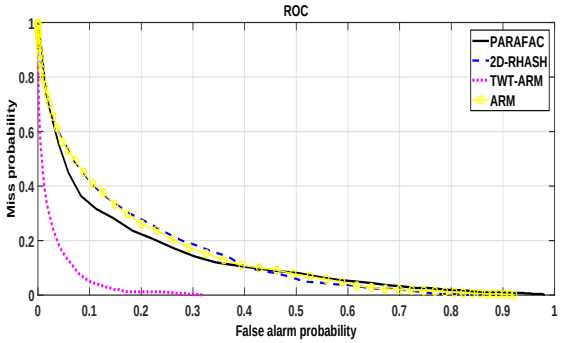


(b) Only 8 frames retained and 8° rotation.

Figure 3.46: Statistical evaluation of video hashing algorithms via the ROC curves under frame-dropping and interpolation followed by rotation attacks.

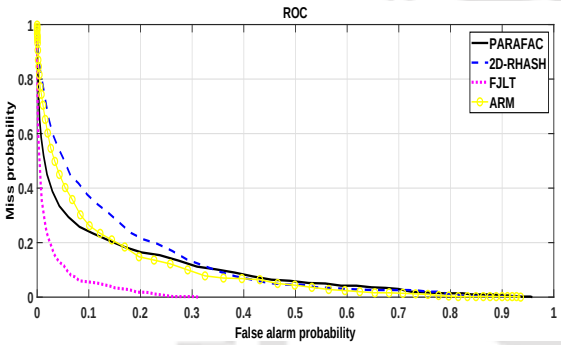


(a) Increase in spatial resolution to 500 × 500, only 32 frames retained and 5° rotation.

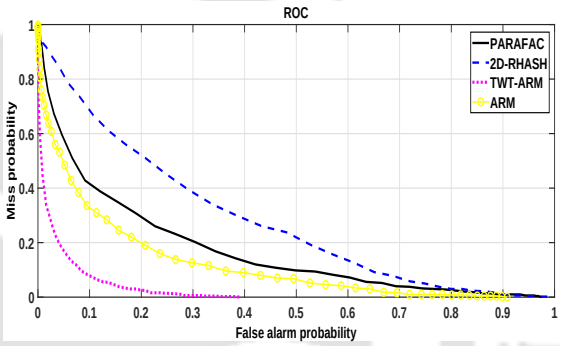


(b) Increase in spatial resolution to 1000 × 1000, only 16 frames retained and 8° rotation.

Figure 3.47: Statistical evaluation of video hashing algorithms via the ROC curves under increased spatial resolution, frame-dropping and interpolation followed by rotation attacks.



(a) Decrease in spatial resolution to 250 × 250, only 32 frames retained and average blurring using 5 × 5 mask.



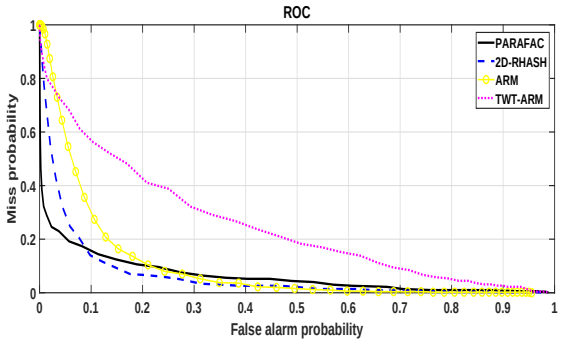
(b) Decrease in spatial resolution to 100 × 100, only 16 frames retained and average blurring using 9 × 9 mask.

Figure 3.48: Statistical evaluation of video hashing algorithms via the ROC curves under decreased spatial resolution, frame-dropping and interpolation followed by average blurring attacks.

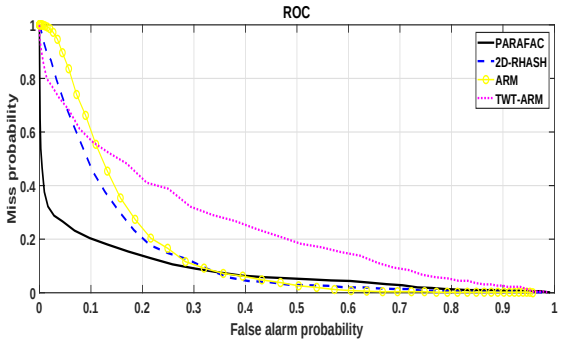
or 1000 × 1000, subjected to frame dropping at regular intervals while retaining only 32 or 16 of the original frames and then interpolated to obtain 64 number of frames, followed by 5° or 8° rotation attacks respectively. The ROC curves for these multiple attacks are shown in Figure 3.47. The area under the ROC curves is smaller for the video hashing algorithm based on the the TWT-ARM. The other video hashing algorithms considered show relatively poor performance.

Decreased spatial resolution, frame-dropping and interpolation followed by average blurring attacks: In this attack, the spatial resolution of each frame was decreased to 250 × 250 or 100 × 100, subjected to frame dropping at regular intervals while retaining only 32 or 16 of the original frames and then interpolated to obtain 64 number of frames, followed by mean blurring using a mask of 5 × 5 or 9 × 9 respectively. The ROC curves for these multiple attacks are shown in Figure 3.48. The area under the ROC curves is smaller for the video hashing algorithm based on the TWT-ARM. The other video hashing algorithms considered show relatively poor performance.

Addition of AWGN, frame insertion and Gaussian blurring attacks: In this attack,



(a) Adding a zero mean AWGN with $\sigma = 81$, 4 unwanted frames inserted and Gaussian blurring using a 5×5 mask.



(b) Adding a zero mean AWGN with $\sigma = 114$, 8 unwanted frames inserted and Gaussian blurring using a 9×9 mask.

Figure 3.49: Statistical evaluation of video hashing algorithms via the ROC curves under addition of AWGN, frame insertion and Gaussian blurring attacks.

the video frames were subjected to zero mean AWGN with $\sigma = 81$ or $\sigma = 114$, insertion of 4 or 8 unwanted frames and the Gaussian filtering with a mask size of 5×5 or 9×9 respectively. The corresponding ROC curves are shown in Figure 3.49. This is a combination of content-preserving attack and the malicious attack. The TWT-ARM based video hashing algorithm is having a relatively larger area under the ROC curves indicating better performance, followed by the video hashing algorithms based on the ARM and the 2D-RASH. The performance of the video hashing algorithm based on the PARAFAC decomposition is perceptually robust and fails to capture the malicious attack.

3.9 Concluding Remarks on TWT-ARM based Perceptual Video Hash

The performance of the the TWT-ARM based video hashing algorithm was found to be robust to the following single image processing attacks: average blurring, Gaussian blurring, addition of AWGN, salt and pepper noise, rotation, cropping, modified brightness, frame dropping and interpolation, modified spatial resolutions, compression, reverse play, and insertion of a logo as the watermark. For most of the multiple attacks, the performance of the proposed video hashing algorithm was found to be comparable to the other video hashing algorithms considered.

From the validation and the performance evaluation of the hash, it can be concluded that the performance of the TWT-ARM based video hashing algorithm is better than the ARM-based video hashing algorithm for most of the typical image processing attacks. It is because in this algorithm the random projection is applied on the low-pass frames containing essential perceptual features of the video rather than directly on the video. But still, it has the scope for improvement when compared to the other video hashing algorithms considered. Furthermore, the computational complexity of the TWT-ARM based video hashing algorithm increases with the increase in the number of video frames.

3.10 Summary of the Chapter

This chapter is summarized as follows.

- The focus of this work was developing a computationally efficient, and simple-to-implement video hashing algorithm and this was achieved by use of Achlioptas's random projections.
- The Achlioptas's random projection matrix consists of +1 and -1 and hence results only in additions and subtractions.
- Two methods were proposed to produce the perceptual hash vector from the input video.
- In one of the methods, the vectorized grey pixels were projected on to the Achlioptas's random basis to generate the compact hash vector.
- This hash based on the ARM was experimentally validated for its desirable properties. The characteristics of the obtained hash were fairly satisfying the desired properties of the perceptual hash.
- This scheme is computationally efficient, and the performance was comparable to the existing video hashing algorithms and the video hashing algorithm based on 3D radial projections.
- In the second method, the TIRFs were generated using the TWT. These TIRFs were vectorized and projected on to the Achlioptas's random basis to generate the compact hash vector.
- The characteristics of this hash improved considerably satisfying the desired properties.
- The evaluation of the algorithm using the ROC curves showed considerable improvement in its performance.
- The computational complexity of the TWT-based video hashing algorithm increases with the increase in the number of video frames.

Chapter 4

Perceptual Video Hashing based on Tucker Decomposition

Contents of the Chapter

Data in the form of three-dimensional arrays occur in a variety of applications such as video compression, video surveillance, live face recognition, video monitoring, video summarization, content-based video retrieval and video tracking. The huge size of the data, memory restrictions and the speed requirements make it difficult to process such data. The alternate way is to apply the dimensionality reduction technique on these three-dimensional arrays and then process them further. The most suitable method to reduce the dimension is by treating these three-dimensional arrays as tensors and then applying the tensor decomposition on these data. The tensor decomposition breaks a large-size tensor into a simpler form of low dimension entities for the extraction of relevant features.

A *tensor* is a multidimensional array. There are two important types of tensor decompositions in the literature [18], [19], [49], [50], namely the *PARAllel FACtor* (PARAFAC) *decomposition* and the *Tucker decomposition*. The PARAFAC decomposition computes the

approximation of a given tensor as a sum of rank-one tensors (to be defined later) while the Tucker decomposition is the generalization of the matrix *singular value decomposition* (SVD).

Tensor decompositions are used for various applications in different fields. It is extensively used for chemical analysis [51] and psychometrics [52]. It is also applied to extend Weiner filter in signal processing [53] and genomic signal processing [54]. It has found important applications in Computer Vision with Tensor Faces [55], for gait recognition [56], watermarking MPEG videos [57], identifying hand-written digits [58, 59], face recognition [60, 59, 61], 3D facial expression recognition [62], determining the foreground region by background subtraction [63], estimating Gaussian mixture models [64] and controller design [65].

In [14, 15], Li and Monga have modelled videos as third-order tensors for perceptual hashing of videos. The authors experimentally validated that this type of modelling captures the spatial and the temporal essence of the video. The multi-linear sub-space projection of tensors, such as low-rank tensor approximations (LRTAs) via PARAFAC decomposition are used for the generation of the perceptual hash from the videos. The authors perform a random 3D tiling of the video through overlapping subtensors and compute LRTAs from these subtensors. Each subtensor was treated as a third-order tensor and then approximated using a rank one tensor. In other words, each third-order tensor is factorized into the outer product of three vectors. These vectors are used to form the perceptual hash. The performance of the algorithm was found to be robust to most of the content-preserving distortions except for the malicious modifications. Further, there is a scope for the generalization of tensor decomposition to improve the performance of the algorithm.

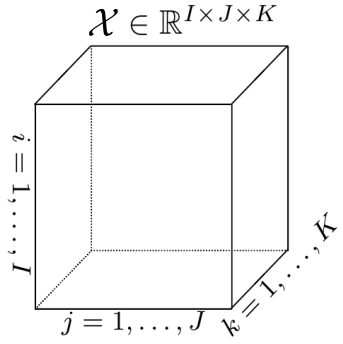
We extend the idea of tensor decomposition in [14, 15] for the generation of perceptual video hashes using the Tucker decomposition [66, 67]. The Tucker decomposition was introduced by Tucker in 1963 [66]. It is also known by few other names like three-mode factor analysis [67], three-mode principal component analysis [68] and higher-order SVD [69]. It decomposes a tensor into a core tensor multiplied by a matrix along each mode [18]. It is a generalized and a powerful tool for the analysis of the multi-way data arrays. It covers the PARAFAC model as a special case and has been applied for the decomposition and the interpretation of multi-way data arrays in many applications [18], [64]. Being a generalized tool, this model also allows the users to select different number of factors along each mode, during the multi-way data array decomposition, aiding better analysis in the case of perceptual video hashing.

The main contributions of this chapter can be enumerated as follows:

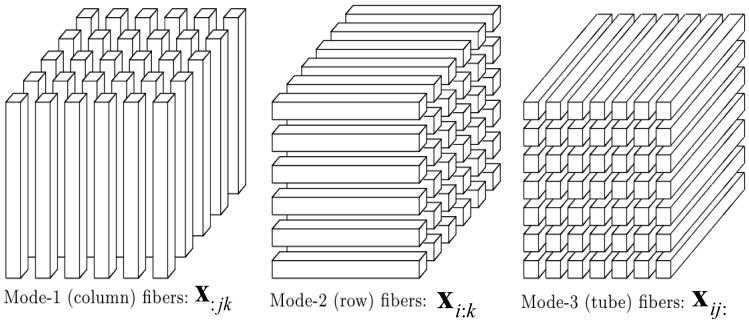
1. A generalized Tucker decomposition method is proposed for the generation of perceptual hashes from the videos.

2. A method is proposed for choosing the suitable number of components in the factor matrices of Tucker decomposition.

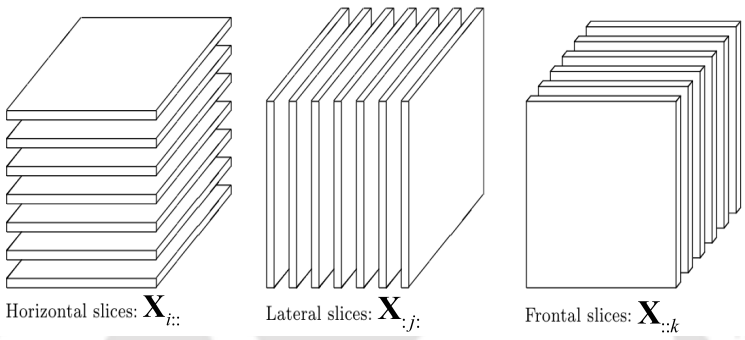
The flow of the chapter is as per the following. The framework for the Tucker decomposition is described in Section 4.1. The selection of parameters for the Tucker decomposition is proposed in Section 4.2. The Tucker decomposition video hashing algorithm based on the Tucker decomposition is presented in Section 4.3. The validation of perceptual hash properties and the performance of the Tucker decomposition based video hashing algorithm is evaluated in



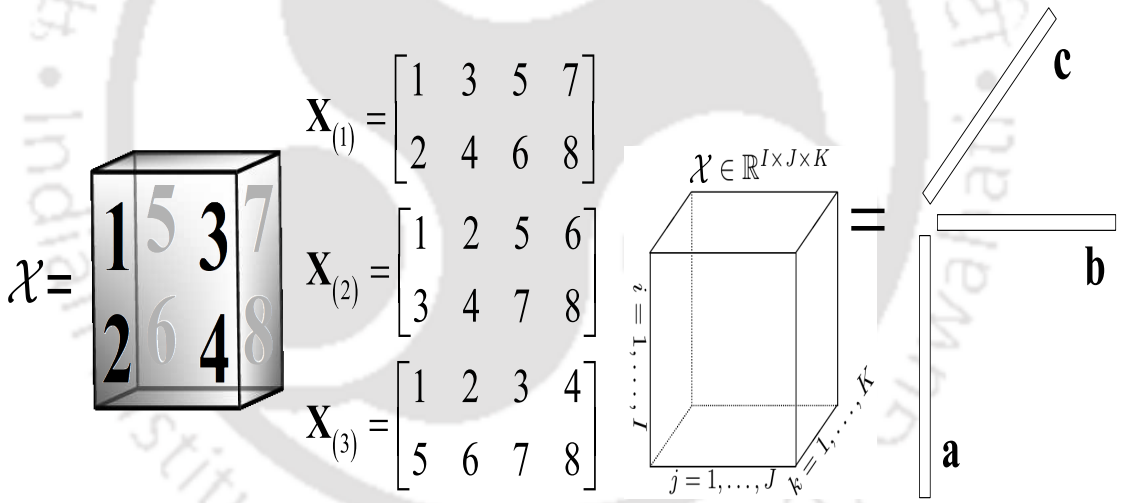
(a) Third-order tensor.



(b) Fibers of a third-order tensor.



(c) Slices of a third-order tensor.



(d) Unfolding of a third-order tensor.

(e) Rank-one third-order tensor.

Figure 4.1: Third-order tensor, fibers, slices, unfolding and rank-one tensor.

the Section 4.4. The comparison in terms of computational complexity of all the video hashing algorithms considered is presented in Section 4.5.

4.1 Mathematical Framework of the Tucker Decomposition

Before introducing the concept of Tucker decomposition the following frequently used terms in the framework of tensor decompositions are defined [18], [19].

The *order* of a tensor is the number of dimensions, also known as *ways* or *orders*. The tensor *fiber* is a one-dimensional fragment of a tensor, obtained by fixing all indices except one. The

tensor *slice* is a two-dimensional section (fragment) of a tensor, obtained by fixing all indices except two. The *unfolding* or *matricization* or *flattening* of a tensor is the process of reordering the elements of a multi-way tensor into a matrix. The mode-n unfolding of a multi-way tensor $\mathcal{X}$ is denoted by $\mathbf{X}_{(n)}$ and arranges the mode-n fibers to be the columns of the resulting matrix. A multi-way tensor of order n is rank-one if it can be written as the outer product of n vectors. If a tensor of order three can be written as an outer product of three vectors, then it is said to be of *rank-one*. Figure 4.1 illustrates all the terms mentioned above for a third-order tensor.

The *rank* of a tensor $\mathcal{X}$, denoted by $rank(\mathcal{X})$, is defined as the smallest number of rank-one tensors that generate $\mathcal{X}$ as their sum. The *n-mode product* of a tensor $\mathcal{G} \in \mathbb{R}^{\lambda \times \mu \times \nu}$ with a matrix $\mathbf{A} \in \mathbb{R}^{\alpha \times \lambda}$, $\mathbf{B} \in \mathbb{R}^{\beta \times \mu}$ and $\mathbf{C} \in \mathbb{R}^{\gamma \times \nu}$ is given by

$$\mathcal{Y} = \mathcal{G} \times_1 \mathbf{A} \Leftrightarrow \mathbf{Y}_{(1)} = \mathbf{A}\mathbf{G}_{(1)} \; ; \quad \mathcal{Y} \in \mathbb{R}^{\alpha \times \mu \times \gamma}$$

$$\mathcal{Y} = \mathcal{G} \times_2 \mathbf{B} \Leftrightarrow \mathbf{Y}_{(2)} = \mathbf{B}\mathbf{G}_{(2)} \; ; \quad \mathcal{Y} \in \mathbb{R}^{\lambda \times \beta \times \gamma}$$

$$\mathcal{Y} = \mathcal{G} \times_3 \mathbf{C} \Leftrightarrow \mathbf{Y}_{(3)} = \mathbf{C}\mathbf{G}_{(3)} \; ; \quad \mathcal{Y} \in \mathbb{R}^{\lambda \times \mu \times \nu}$$

Given a large-sized tensor, the Tucker decomposition computes its best approximation in-terms of a small-sized tensor called the *core tensor*, multiplied by a *factor matrix* along each mode [18]. It is a flexible technique for performing the decomposition of n-way data arrays into a lower dimensional space. In the present case, we model the video as an order three tensor. Therefore, we would only be dealing with the decomposition of three-way data. If the number of components in each mode is equal, then the Tucker decomposition reduces to the PARAFAC decomposition. Therefore, the Tucker decomposition can be viewed as a generalization of the PARAFAC decomposition.

4.1.1 Approximation of a Third-order Tensor using the Tucker decomposition.

Given a third-order tensor $\mathcal{X} \in \mathbb{R}^{\alpha \times \beta \times \gamma}$ and three positive indices $\{\lambda, \mu, \nu\} << \{\alpha, \beta, \gamma\}$, the Tucker decomposition determines the core tensor $\mathcal{G} \in \mathbb{R}^{\lambda \times \mu \times \nu}$ and *three component matrices* or *loading matrices* $\mathbf{A} \in \mathbb{R}^{\alpha \times \lambda}$, $\mathbf{B} \in \mathbb{R}^{\beta \times \mu}$ and $\mathbf{C} \in \mathbb{R}^{\gamma \times \nu}$ [19], [18], [70]. The common notations used for defining the Tucker decomposition are given in Table 4.1. Mathematically,

$$\mathcal{X} = \sum_{r=1}^{\lambda} \sum_{p=1}^{\mu} \sum_{q=1}^{\nu} g(r,p,q) \left(\mathbf{a}_r \circ \mathbf{b}_p \circ \mathbf{c}_q\right) + \mathbf{E} \tag{4.1}$$

where $\mathbf{a}_r$, $\mathbf{b}_p$, and $\mathbf{c}_q$ are the column vectors of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, respectively;

$$\mathbf{E} = \mathcal{X} - \hat{\mathcal{X}}$$

and $\hat{\mathcal{X}}$ is given by

$$\hat{\mathcal{X}} = \sum_{r=1}^{\lambda} \sum_{p=1}^{\mu} \sum_{q=1}^{\nu} g(r,p,q) \left(\mathbf{a}_r \circ \mathbf{b}_p \circ \mathbf{c}_q\right) = [\![\mathcal{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \tag{4.2}$$

Table 4.1: Notations involved in defining the Tucker decomposition.

Notation	Description
$\mathcal{X}$	Third-order tensor
$\hat{\mathcal{X}}$	An approximation for $\mathcal{X}$
λ, μ, ν	Number of components along each mode
$\mathcal{G}$	Core tensor
$g(i,j,k)$	Elements of $\mathcal{G}$
$\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$	Component matrices
$\mathbf{a}_r$, $\mathbf{b}_p$, and $\mathbf{c}_q$	Column vectors of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$.
$a(i,r), b(j,p)$ and $c(k,q)$	Elements of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$
$\mathbf{E}$	Residual or error tensor
$e(i,j,k)$	Elements of $\mathbf{E}$
$[\![\ ]\!]$	Concise representation of the tensor decomposition
$\circ$	Outer product between the two vectors
$\|\ \|_F$	Frobenius norm of a tensor

If $x(i,j,k)$ represents an element of the tensor $\mathcal{X}$, then the Tucker decomposition in element-wise form is given by

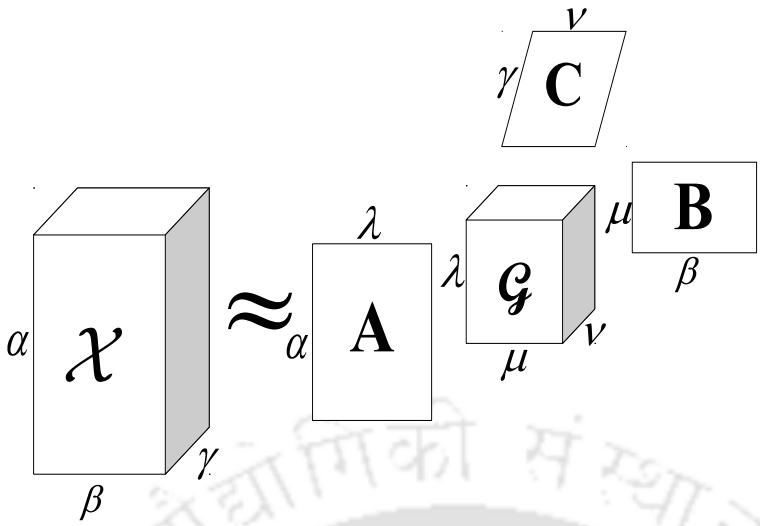
$$x(i,j,k) = \sum_{r=1}^{\lambda}\sum_{p=1}^{\mu}\sum_{q=1}^{\nu} g(r,p,q)a(i,r)b(j,p)c(k,q) + e(i,j,k) \tag{4.3}$$

The Tucker decomposition of a three-way array (also called Tucker3 model) is shown in Figure 4.2(a).

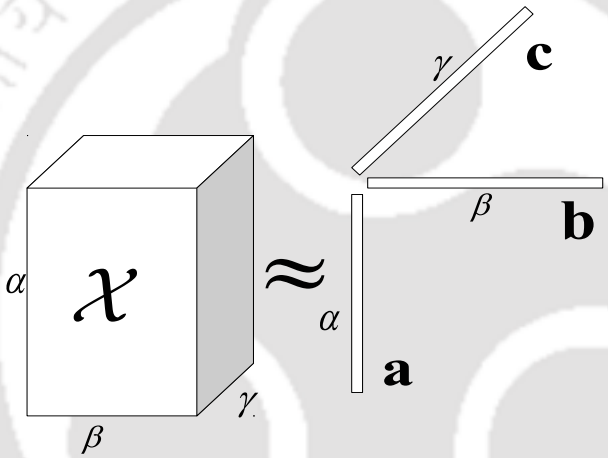
The elements of $\mathcal{G}$ give different weights to the product of $a(i,r), b(j,p)$ and $c(k,q)$. The parameters λ, μ and ν in Equation 4.1 indicate the number of components (i.e., columns) used in the Tucker decomposition in each direction. Thus, the Tucker3 model is a weighted sum of the outer product of three vectors (factors) stored as columns of component matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. The weights are the corresponding elements of core tensor $\mathcal{G}$. If $\lambda = \mu = \nu$, then the Tucker decomposition reduces to PARAFAC decomposition. The PARAFAC decomposition is shown in Figure 4.2(b).

In [69], the authors obtained the Tucker decomposition of the tensor by extending the matrix SVD, called the *higher-order SVD* (HOSVD). The HOSVD fits each individual mode optimally without taking into account interactions among the modes. The drawback of this technique is that it does not obtain the best fit for a given tensor. In [71], the authors proposed an efficient way of calculating the loading matrices and called it the *Higher-order Orthogonal Iteration* (HOOI) method. The HOOI method is computationally demanding than HOSVD. The details of the method are as follows.

Consider a third-order tensor $\mathcal{X} \in \mathbb{R}^{\alpha \times \beta \times \gamma}$ and three positive indices $\{\lambda, \mu, \nu\} << \{\alpha, \beta, \gamma\}$. The goal of the Tucker decomposition is to find the best approximation $\hat{\mathcal{X}}$ in some meaningful



(a) The Tucker decomposition.



(b) The PARAFAC decomposition.

Figure 4.2: Pictorial representation of the tensor decomposition of a three-way array using Tucker and PARAFAC decomposition techniques.

sense. Generally, the problem is expressed as

$$\underset{[\mathcal{G};\mathbf{A},\mathbf{B},\mathbf{C}]}{\text{Min}} \left\| \mathcal{X} - \hat{\mathcal{X}} \right\|_F \tag{4.4}$$

such that the matrices **A**, **B** and **C** are column-wise orthogonal and also orthogonal to each other; the core tensor is also all-orthogonal (the rows, the columns and the matrices are all mutually orthogonal) and its matrices are ordered with a decreasing Frobenius norm. The orthogonality and decreasing Frobenius norm conditions are required to keep the whole procedure similar to the SVD.

If $\mathcal{X}$ is an $i \times j \times k$ third-order tensor, then it can be unfolded to a two-way array $\mathbf{X}_{(1)}$ of dimensions $i \times jk$. Similarly we can get unfolded two-way arrays $\mathbf{X}_{(2)}$ and $\mathbf{X}_{(3)}$ of dimensions $j \times ki$ and $k \times ij$ respectively. **A** is calculated by performing the SVD on the matrix obtained by flattening or matricization of $\mathcal{X}$. In the matricized forms, the Tucker approximation can be

written as

$$\mathbf{X}_{(1)} \approx \mathbf{A}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T \tag{4.5}$$

$$\mathbf{X}_{(2)} \approx \mathbf{B}\mathbf{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^T \tag{4.6}$$

and

$$\mathbf{X}_{(3)} \approx \mathbf{C}\mathbf{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^T \tag{4.7}$$

where '$\otimes$' denotes the Kronecker product between the two matrices; $\mathbf{G}_{(1)}, \mathbf{G}_{(2)}$ and $\mathbf{G}_{(3)}$ represent the slice of the core tensor $\mathcal{G}$ parallel to the ij, jk and ki planes, respectively.

The approximation in Equation 4.5 can be solved for $\hat{\mathbf{A}}$ using the alternating least squares (ALS) approach as follows.

To start with, the core tensor $\mathcal{G}$ and the loading matrices $\mathbf{B}$ and $\mathbf{C}$ are randomly initialized. With these values of $\mathcal{G}$, $\mathbf{B}$ and $\mathbf{C}$ the ALS solution $\hat{\mathbf{A}}$ is obtained by solving

$$\min_{\hat{\mathbf{A}}} \left\| \mathbf{X}_{(1)} - \hat{\mathbf{A}}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T \right\|_F \tag{4.8}$$

The least squares estimate $\hat{\mathbf{A}}$ is given as

$$\hat{\mathbf{A}} = \mathbf{X}_{(1)}\mathbf{G}_{(1)}\left[(\mathbf{C} \otimes \mathbf{B})^T\right]^{\dagger} \tag{4.9}$$

where '$\dagger$' denotes the pseudo-inverse operation.

$\hat{\mathbf{A}}$ is then substituted for $\mathbf{A}$ in Equation 4.6 and the least squares estimate of $\mathbf{B}$ denoted by $\hat{\mathbf{B}}$ is calculated. Using $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ for $\mathbf{A}$ and $\mathbf{B}$, the least squares estimate of $\mathbf{C}$ denoted by $\hat{\mathbf{C}}$ is calculated using Equation 4.7. Once the $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$ are available, $\mathcal{G}$ is updated as.

$$g(i,j,k) = \sum_{r=1}^{\lambda}\sum_{p=1}^{\mu}\sum_{q=1}^{\nu} x(r,p,q)\hat{a}(i,r)\hat{b}(j,p)\hat{c}(k,q) \tag{4.10}$$

where $\hat{a}(i,r), \hat{b}(j,p)$, and $\hat{c}(k,q)$ are the elements of $\hat{\mathbf{A}}, \hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$, respectively. The above procedure of finding least squares estimate $\hat{\mathbf{A}}, \hat{\mathbf{B}}$, $\hat{\mathbf{C}}$ and updating $\mathcal{G}$ is repeated until either of the conditions $\left\| \mathcal{X} - \hat{\mathcal{X}} \right\|_F \leq Th$, where Th is a small value or the number of iterations $= iter_{max}$ is satisfied. The threshold, Th is a small scalar decided by the trial and error method. $iter_{max}$ is chosen to come out of the loop if the best fit is not obtained. The computation steps of the Tucker decomposition is described in Algorithm 4.1.

The main advantage of the Tucker decomposition lies in its flexibility in choosing a different number of factors corresponding to each of the modes. This flexibility is absent in the PARAFAC decomposition as it approximates the video as the linear combination of rank-one tensors. Hence, the Tucker decomposition can be made to capture the variations in each mode. The advantage of the Tucker decomposition is also its weakness because it is often challenging to determine a unique representation for any given tensor and identify the dominant relationships between the components.

We propose a method to find the suitable number of components (i.e., λ, μ and ν) in the factor matrices of the Tucker decomposition, by finding the maximum variation of the luminance values of the pixels along each direction. As the video is treated as an order 3 tensor, variations in the luminance values of the pixels along the spatial and the temporal direction are to be captured to determine the pertinent values of λ, μ and ν.

Algorithm 4.1 *Computation of the Tucker decomposition using the HOOI Method.*
Input: $\mathcal{X}, \lambda, \mu, \nu, Th$ and $iter_{max}$.

1. Initialize $\mathcal{G}$, $\mathbf{B}$, and $\mathbf{C}$ with random values; $iter = 1$.
2. Fix $\mathcal{G}$, $\mathbf{B}$ and $\mathbf{C}$ to solve for $\mathbf{A}$ using the alternating least squares approach.

$$\min_{\hat{\mathbf{A}}} \left\| \mathbf{X}_{(1)} - \hat{\mathbf{A}}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T \right\|_F$$

   to get $\hat{\mathbf{A}} = \mathbf{X}_{(1)}\mathbf{G}_{(1)}\left[(\mathbf{C} \otimes \mathbf{B})^T\right]^\dagger$
3. $A \leftarrow \hat{A}$
4. Solve for $\mathbf{B}$ using

$$\min_{\hat{\mathbf{B}}} \left\| \mathbf{X}_{(2)} - \hat{\mathbf{B}}\mathbf{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^T \right\|_F$$

   to get $\hat{\mathbf{B}} = \mathbf{X}_{(2)}\mathbf{G}_{(2)}\left[(\mathbf{C} \otimes \mathbf{A})^T\right]^\dagger$
5. $B \leftarrow \hat{B}$
6. Solve for $\mathbf{C}$ using

$$\min_{\hat{\mathbf{C}}} \left\| \mathbf{X}_{(3)} - \hat{\mathbf{C}}\mathbf{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^T \right\|_F$$

   to get $\hat{\mathbf{C}} = \mathbf{X}_{(3)}\mathbf{G}_{(3)}\left[(\mathbf{B} \otimes \mathbf{A})^T\right]^\dagger$
7. $C \leftarrow \hat{C}$
8. Update each element $g(i,j,k)$ of $\mathcal{G}$ as

$$g(i,j,k) = \sum_{r=1}^{\lambda}\sum_{p=1}^{\mu}\sum_{q=1}^{\nu} x(r,p,q)a(i,r)b(j,p)c(k,q)$$

9. Obtain $\hat{\mathcal{X}} = \sum_{r=1}^{\lambda}\sum_{p=1}^{\mu}\sum_{q=1}^{\nu} g(r,p,q)\,(\mathbf{a}_r \circ \mathbf{b}_p \circ \mathbf{c}_q)$
10. $iter = iter + 1$
11. Repeat step 2 to step 10 until either $\left\|\mathcal{X} - \hat{\mathcal{X}}\right\|_F \leq Th$ or $iter = iter_{max}$.

Output: $\hat{\mathcal{X}}$

4.2 Selection of Parameters: λ, μ and ν

The authors in [72] demonstrate the following about the Tucker decomposition through exhaustive experimentations.

1. The Tucker decomposition obtains a good approximation of the original tensor with high perceptual quality. This results in a viable technique for compressed volume representation.

2. Given a tensor $\mathcal{X} \in \mathbb{R}^{U \times U \times U}$, for a high-quality data reduction, the rank of the Tucker decomposition is less than or equal to $\frac{U}{2}$.

3. The PARAFAC decomposition requires a large number of ranks to achieve the accuracy of the Tucker decomposition.

4. The Tucker decomposition has an excellent performance with respect to other Tensor approximations in terms of both quality and time.

Furthermore, in [14, 15], the authors used rank-1 $\{\lambda = \mu = \nu = 1\}$ PARAFAC decomposition to construct the perceptual video hash. The hash showed a good performance to most of the common image processing attacks, but there is still a scope for improvement.

Our objective is not to reconstruct the video tensor but to capture the essential perceptual features from the original tensor and obtain a robust perceptual video hash. From the above discussion, we argue that a very low-rank Tucker decomposition is sufficient to capture the essential features from the original tensor. In addition, the Tucker decomposition has the flexibility of choosing different values of $\{\lambda, \mu, \nu\}$. Therefore, the objective is to find a suitable combination(s) of small values of $\{\lambda, \mu, \nu\}$ such that the perceptual features are captured more relevantly and construct a robust perceptual video hash from a very low-rank Tucker decomposition. It has to be noted that if we increase one of the values of $\{\lambda, \mu, \nu\}$, then the resultant decomposition can capture important features along the corresponding direction and hence results in a better fit.

To achieve the objective mentioned above, it is necessary to find the direction of the variation in the luminance value of the pixels from the video data. Therefore, we need to analyse the variation in the luminance value of the pixels along each of the 3 directions of a video.

To quantify the variation in the pixel luminance in x, y and z directions, we introduce three variance parameters δ_x, δ_y and δ_z.

Suppose $\mathcal{X}(i, j, k)$, $i = 1, \ldots, \alpha$; $j = 1, \ldots, \beta$; $k = 1, \ldots, \gamma$; denotes the video. Then, δ_x, δ_y and δ_z are given by

$$\delta_x = \sum_{k=1}^{\gamma} \sum_{i=1}^{\alpha} Var\left(\mathcal{X}(i, :, k)\right) \tag{4.11}$$

$$\delta_y = \sum_{k=1}^{\gamma} \sum_{j=1}^{\beta} Var\left(\mathcal{X}(:, j, k)\right) \tag{4.12}$$

and

$$\delta_z = \sum_{i=1}^{\alpha} \sum_{j=1}^{\beta} Var\left(\mathcal{X}(i, j, :)\right) \tag{4.13}$$

where ':' denotes all the pixels along a particular direction and 'Var' denotes the variance. Once δ_x, δ_y and δ_z are obtained, the number of factors in each mode of the Tucker decomposition can be chosen as per the following discussion.

In our formulation, the Tucker decomposition on a tensor $\mathcal{X} \in \mathbb{R}^{U \times U \times U}$, would result in a hash length of $k = (U \times \lambda) + (U \times \mu) + (U \times \nu) = U(\lambda + \mu + \nu)$ (procedure to be discussed later).

For a fixed U, the value of $\lambda + \mu + \nu$ will determine the hash length. The value of δ_x, δ_y and δ_z will in turn determine the value of λ, μ and ν. If there is more variation in the pixel luminance along a particular direction, then a higher value of corresponding λ or μ or ν is selected to capture the excess variations. Similarly less variation leads to a parameter value of 1 along that direction. If $k = 3U$ then the only possible combination of $\{\lambda, \mu, \nu\}$ is $\lambda = \mu = \nu = 1$. If $k = 4U$, then the possible combination of $\{\lambda, \mu, \nu\}$ are $\{1, 1, 2\}$, $\{1, 2, 1\}$ and $\{2, 1, 1\}$. Thus, we conclude that for a fixed value of U and the hash length k, the value of δ_x, δ_y and δ_z will determine the value of λ, μ and ν.

4.3 Proposed Video Hashing Algorithm

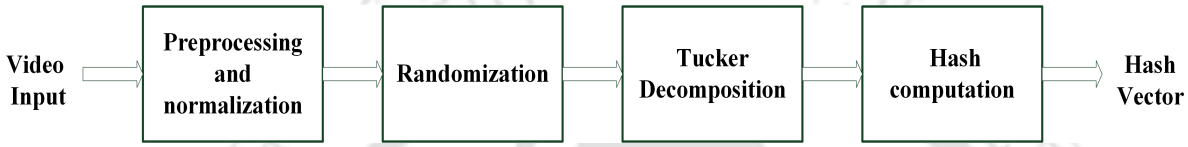


Figure 4.3: The block diagram representation for the extraction of perceptual video hash based on the Tucker decomposition.

A generalized block diagram of the proposed video hashing algorithm based on the Tucker decomposition is shown in Figure 4.3. The steps involved in obtaining a hash for a video are as follows: *pre-processing and normalization, randomization, Tucker decomposition and hash computation*. A brief description of the different steps are as follows:

(i) **Preprocessing and normalization**: To account for the effect of the variations in video's spatial resolution and the number of frames on the output hash, the input videos are normalized to $X \times X \times X$ via temporal sub-sampling and spatial re-sizing. Let the normalized video be represented by $\mathcal{V}_{norm}$.

(ii) **Randomization**: N number of randomly overlapping 3D subtensors of video of predefined size $U \times U \times U$, such that $U < X$ are chosen. Let $\mathcal{V}_i$, $i = 1, 2, \ldots, N$, be the randomly overlapping 3D subtensors. A secret key K is used for the random selection of the subtensors. The variation in secret key leads to the selection of different subtensors and eventually a different hash and thus ensuring that the hash extraction process is secured. N is chosen by trial and error such that it approximately covers $\mathcal{V}_{norm}$.

(iii) **Tucker decomposition**: The values δ_x, δ_y and δ_z are calculated for all the videos in the database. For each video, the maximum of δ_x, δ_y and δ_z are determined. For a given U and k, the number of factors in the component matrices λ, μ and ν is decided using the procedure described in the Section 4.2. To keep the same combination of λ, μ and ν orient the video appropriately. Each subtensor $\mathcal{X}_i$, $i = 1, 2, ..., N$ of size $U \times U \times U$ is considered as an order 3 tensor and decomposed using low-rank Tucker decomposition described in Algorithm 4.1. Each subtensor $\mathcal{X}_i$ is effectively represented by three loading matrices $\mathbf{A}_i$, $\mathbf{B}_i$ and $\mathbf{C}_i$ using the Tucker decomposition.

(iv) **Hash Computation:** After the Tucker decomposition, the final step is to compute the hash. The component matrices $\mathbf{A}_i$, $\mathbf{B}_i$ and $\mathbf{C}_i$ obtained are of sizes $U \times \lambda$, $U \times \mu$ and $U \times \nu$ respectively. They are concatenated column wise to form a vector $\mathbf{h}_i$ of length $k = (U \times \lambda) + (U \times \mu) + (U \times \nu) = U(\lambda + \mu + \nu)$ for each $\mathcal{V}_i$, $i = 1, 2, ..., N$. An intermediate hash vector $\mathbf{h}'$ is obtained by performing the arithmetic averaging on all the $\mathbf{h}_i$s corresponding to all the subtensors using, $\mathbf{h}' = \frac{\sum_{i=1}^{N} \mathbf{h}_i}{N}$. For an even hash length, the median, $\xi = \left(\frac{h'\left(\frac{k}{2}\right) + h'\left(\frac{k}{2}\right)+1}{2} \right)$ is calculated after rank ordering $\mathbf{h}'$. The elements of $\mathbf{h}'$ are binarized using ξ as the threshold to obtain the final hash vector, $\mathbf{h} = [h_1\, h_2 \ldots h_k]^T$ where

$$h_j = \begin{cases} 0; & h'_j < \xi \\ 1; & h'_j \geq \xi \end{cases}, \; j = 1, 2, \ldots, k.$$

The hash generation process is described in the Algorithm 4.2.

4.4 Simulation Results and Discussion

The desirable properties of the perceptual hash are validated through a number of experiments. The performance of the video hashing algorithms are evaluated using ROC curves. The details of the database are tabulated in Table 2.1 and discussed in Chapter 2. The input video, V_{in} is preprocessed and normalized to $\mathcal{V}_{norm} \in \mathbb{R}^{64 \times 64 \times 64}$. For computational simplicity, 7 randomly located subtensors $\mathcal{V}_i$, $i = 1, 2, ..., 7$, each of size $32 \times 32 \times 32$ are selected using a secret key K, such that it covers $\mathcal{V}_{norm}$. Using the method of trial and error, the size of each subtensor was decided. To have a fair comparison with other algorithms, a uniform hash length of 128 is considered. To keep the same combination of λ, μ and ν the input video is oriented appropriately. The Tucker decomposition and the PARAFAC decomposition are implemented using the MATLAB tensor toolbox downloaded from [73].

4.4.1 Experimental Verification for the Selection of Parameters: λ, μ and ν

Section 4.1 noted that the Tucker decomposition has the flexibility of choosing the number of factors along each mode, this leads to a better decomposition technique and thus a method was proposed in Section 4.2 to choose the number of factors in each mode. This flexibility is absent in the PARAFAC decomposition.

The values δ_x, δ_y and δ_z were calculated for the following test sequences, namely '*clip306.mp4*' and '*clip467.mp4*' and the results are tabulated in Table 4.2. For the first test sequence $\delta_y > \delta_x > \delta_z$ which suggest $\mu > \lambda > \nu$ in the Tucker decomposition for better visual representation of the sequence. Similarly for the second test sequence $\delta_y > \delta_z > \delta_x$ which suggest $\mu > \nu > \lambda$ in the Tucker decomposition for better visual representation of the sequence. Table 4.2 shows the PSNR values for the reconstructed test sequences using various Tucker configurations by keeping the original normalized test sequence as the reference. From this table we conclude that the values of λ, μ and ν in the Tucker decomposition can be decided using δ_x, δ_y and δ_z.

Algorithm 4.2 *Proposed video hashing algorithm based on the Tucker decomposition.*

1: **Input**:

- Video input V_{in}, k(even).
- Algorithm parameters: X, U, K and N.

2: **Preprocessing and Normalization**:

- Convert V_{in} (which is generally in RGB format) into a sequence of grey-level frames, V_{grey}.
- Normalize V_{grey} via temporal sub-sampling and spatial re-sizing to obtain $\mathcal{V}_{norm}$ of size $X \times X \times X$.

3: **Randomization**: Depending on secret key K, randomly select N overlapping 3D subtensors $\mathcal{V}_i$, $i = 1, 2, ..., N$, each of size $U \times U \times U$.

4: **Tucker decomposition**

- Calculate δ_x, δ_y and δ_z for all the videos in the database. For each video, determine the maximum of δ_x, δ_y and δ_z.
- Given U and k, determine the appropriate values of λ, μ and ν.
- Apply the Tucker decomposition to each $\mathcal{X}_i$ s for chosen values of λ, μ and ν. $\mathcal{X}_i \approx [\![\mathcal{G}_i; \mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i]\!]$; $\mathbf{A}_i \in \mathbb{R}^{U \times \lambda}, \mathbf{B}_i \in \mathbb{R}^{U \times \mu}$, $\mathbf{C}_i \in \mathbb{R}^{U \times \nu}$.

5: **Hash computation**

- Concatenate the columns of factor matrices $\mathbf{A}_i$, $\mathbf{B}_i$ and $\mathbf{C}_i$ to form a vector $\mathbf{h}_i \in \mathbb{R}^{U \times \lambda + U \times \mu + U \times \nu}$; $k = U(\lambda + \mu + \nu)$.
- Construct the matrix
$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_N \end{bmatrix}.$$
- Generate the intermediate hash vector $\mathbf{h}' = \frac{\sum_{i=1}^{N} \mathbf{h}_i}{N}$, where $\mathbf{h}_i$ is the i^{th} column of the matrix $\mathbf{H}$.
- Rank order $\mathbf{h}'$ to obtain $\mathbf{h}'' = [h'_{(1)}\, h'_{(2)} \ldots h'_{(k)}]^T$.
- Median, $\xi = \left(\frac{h'\left(\frac{k}{2}\right) + h'\left(\frac{k}{2}\right)+1}{2} \right)$.
- Binarize the elements of $\mathbf{h}'$ to obtain $\mathbf{h} = [h_1\, h_2 \ldots h_k]^T$ where
$$h_j = \begin{cases} 0; & h'_j < \xi \\ 1; & h'_j \geq \xi \end{cases}, \; j = 1, 2, \ldots, k.$$

6: **Output**

- A hash vector $\mathbf{h}$

For computational simplicity we choose uniform values of λ, μ and ν for the Tucker decomposition of the videos in the entire video database. To decide the uniform values of λ, μ and ν, δ_x, δ_y and δ_z were calculated for all the videos in the database. For each video, the maximum of δ_x, δ_y and δ_z was determined. The number of times δ_x, δ_y and δ_z are maximum were $153, 579$ and 268 respectively. This indicates that, on the average, the variation in the pixel values within a frame along y direction is maximum and the variation is minimum along the x direction for the database chosen. In our formulation, $U = 32$ and $k = 128$. Therefore, $128 = 32(\lambda + \mu + \nu)$ and the admissible combinations of $\{\lambda, \mu, \nu\}$ are $\{1, 1, 2\}$, $\{1, 2, 1\}$ and $\{2, 1, 1\}$. As we need higher order to preserve the excess variation in the pixel luminance we choose $\lambda = 1, \mu = 2$ and $\nu = 1$ throughout the experiments.

A comparison of the PSNRs of the Tucker approximations for the entire database with the admissible values of λ, μ and ν is shown in Table 4.3. An input video $\mathcal{V}_{in}$ is preprocessed and normalized to a size of $64 \times 64 \times 64$ to obtain $\mathcal{V}_{norm}$. The Tucker decomposition is applied on $\mathcal{V}_{norm}$ by considering $\lambda = 1, \mu = 1$ and $\nu = 1$ to obtain the core tensor $\mathcal{G}_{\lambda=1,\mu=1,\nu=1}$ and the factor matrices $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}_{\lambda=1,\mu=1,\nu=1}$. The approximate version of $\mathcal{V}_{norm}$ is reconstructed back using $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}_{\lambda=1,\mu=1,\nu=1}$ and $\mathcal{G}_{\lambda=1,\mu=1,\nu=1}$ to obtain $\hat{\mathcal{V}}_{norm_{\lambda=1,\mu=1,\nu=1}}$. The PSNR for $\hat{\mathcal{V}}_{norm_{\lambda=1,\mu=1,\nu=1}}$ is calculated by keeping $\mathcal{V}_{norm}$ as reference. Next, the Tucker decomposition is applied on $\mathcal{V}_{norm}$ by choosing $\lambda = 1, \mu = 1$ and $\nu = 2$ to obtain the corresponding core tensor $\mathcal{G}_{\lambda=1,\mu=1,\nu=2}$ and the factor matrices $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}_{\lambda=1,\mu=1,\nu=2}$. $\hat{\mathcal{V}}_{norm_{\lambda=1,\mu=1,\nu=2}}$ is reconstructed back from $\mathcal{G}_{\lambda=1,\mu=1,\nu=2}$ and $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}_{\lambda=1,\mu=1,\nu=2}$. The PSNR for $\hat{\mathcal{V}}_{norm_{\lambda=1,\mu=1,\nu=2}}$ is calculated by keeping $\mathcal{V}_{norm}$ as reference. The same procedure is carried out for the other values of λ, μ and ν. The entire process is repeated for all the remaining videos in the database. All the PSNR values obtained for the entire video database by considering $\lambda = 1, \mu = 1$ and $\nu = 1$ during the Tucker decomposition are averaged to obtain the mean PSNR value. Similarly the other mean PSNR values are calculated from the other PSNR values obtained for the entire video database by considering different values of λ, μ and ν. The results are tabulated in Table 4.3. From Table 4.3, it can be observed that the highest PSNR value of 28.21dB is obtained for $\{\lambda = 1, \mu = 2, \nu = 1\}$ followed by $\{\lambda = 2, \mu = 1, \nu = 1\}$, $\{\lambda = 1, \mu = 1, \nu = 2\}$ and $\{\lambda = 1, \mu = 1, \nu = 1\}$.

Thus, for a given U and k, among all the values of $\{\lambda, \mu, \nu\}$ considered, the values of $\{\lambda = 1, \mu = 2, \nu = 1\}$ result in a hash length of 128 with relatively satisfactory performance. To further justify this combination of $\{\lambda, \mu, \nu\}$, the Tucker decomposition is applied on "akiyo_cif" video with $\{\lambda = 1, \mu = 2, \nu = 1\}$ and reconstructed back including the residuals. The first frame of the normalized input video and the approximated video is shown in Figure 4.5.

4.4.2 Hash Validation

Experiments were conducted to validate the unpredictability, the visual fragility and the perceptual robustness properties of the hash generated using the Tucker decomposition based video hashing algorithm. 224 videos from the database and 1000 secret keys are considered to validate the proposed hash. The NHD was the distance metric used to compare the hash values of the videos. The histograms of the NHD to evaluate the important desirable properties of the hash are shown in Figure 4.6. The validations are described below.

(a) clip306.

(b) clip467.

Figure 4.4: Snapshot of the sample test sequences considered for the selection of λ, μ and ν.

Table 4.2: To validate the dependencies of λ, μ and ν on δ_x, δ_y and δ_z.

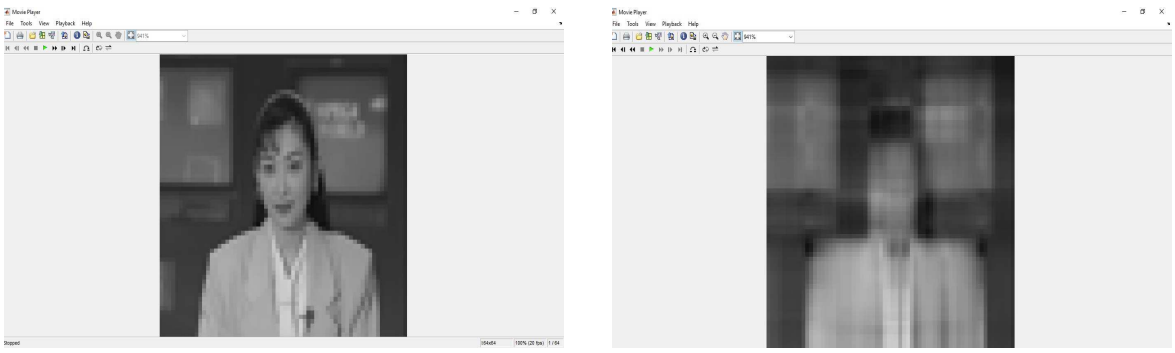
Tucker Configurations	**PSNR (in dB) for clip306** $\{\delta_x, \delta_y, \delta_z\} = \{20.339, 32.176, 24.169\} \times 10^5$	**PSNR (in dB) for clip467** $\{\delta_x, \delta_y, \delta_z\} = \{46.556, 142.760, 42.083\} \times 10^5$
111	16.11	28.57
112	16.24	28.73
121	16.57	29.53
211	16.18	29.06
122	16.86	29.96
212	16.38	29.42
221	16.78	30.60
113	16.28	28.75
131	16.75	29.76
311	16.20	29.13

Table 4.3: Validation of the choice of number of factors along the each mode of the Tucker decomposition.

Tucker configuration	**Mean PSNR value** (in dB)
111	25.40
112	26.58
121	28.21
211	27.34

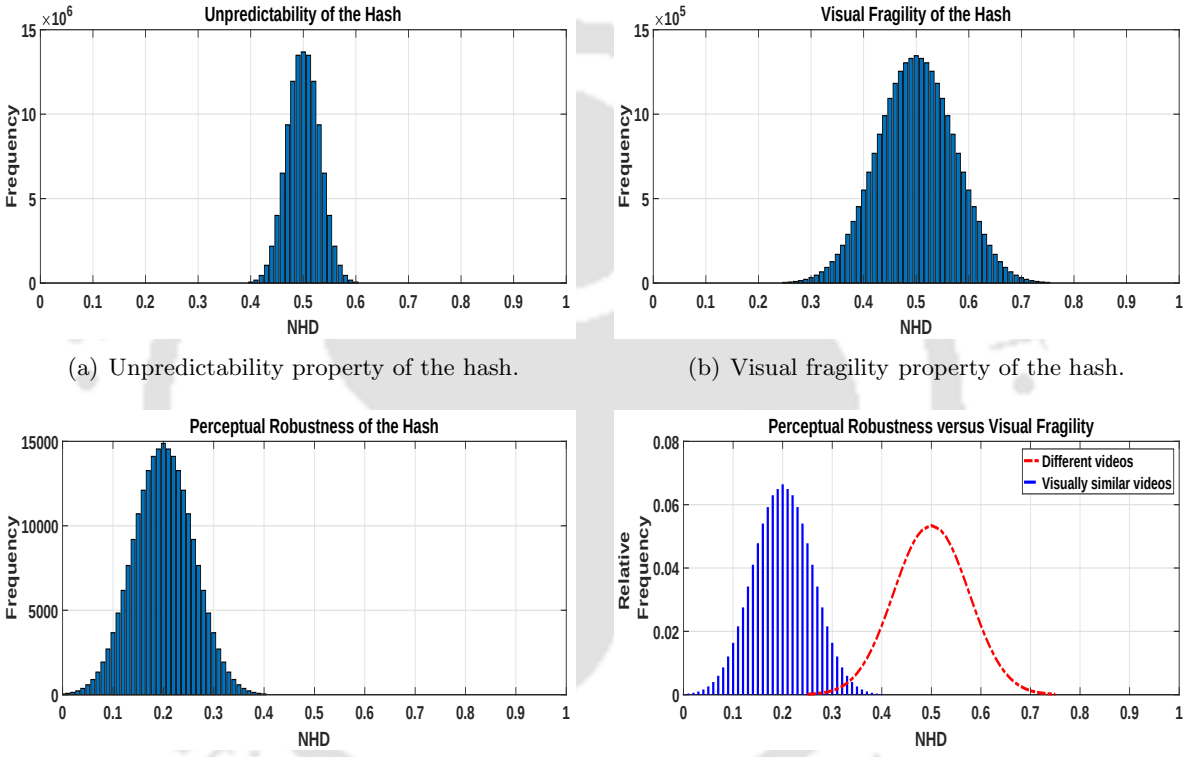
(i) *To validate the unpredictability property*

128-bit hashes were generated for a test sequence using 1000 different secret keys, and then, the NHD was measured between every possible pair of the hashes. The procedure was repeated for the remaining 223 test sequences. The histogram of the NHD values for this experiment is shown in Figure 4.6(a). The figure shows that the NHD measured between the hashes in this case approximately follows a Gaussian distribution with the mean 0.5. A narrow-spread distribution centred around 0.5 implies that the perceptual hash closely satisfies the unpredictability property given in Equation 1.5. Comparing Figure 4.6(a) with Figure 3.27(a) we can conclude that the perceptual hash based on



(a) Normalized input video. (b) Approximated video.

Figure 4.5: Normalized input video and the approximated video using Tucker decomposition.



(a) Unpredictability property of the hash. (b) Visual fragility property of the hash.

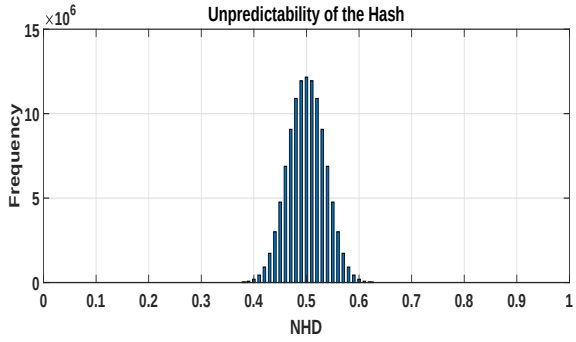
(c) Perceptual robustness property of the hash. (d) Perceptual robustness property versus the visual fragility of the hash.

Figure 4.6: The histogram of normalized Hamming distance to evaluate the important desirable properties of the hash generated using the Tucker decomposition based video hashing algorithm.

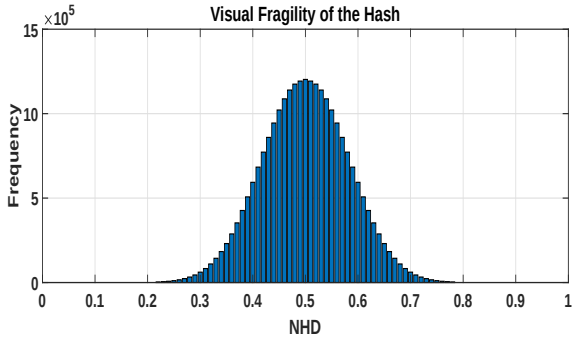
the Tucker decomposition satisfies the unpredictability property more closely than the perceptual hash based on the TWT-ARM.

(ii) *To validate the visual fragility property*

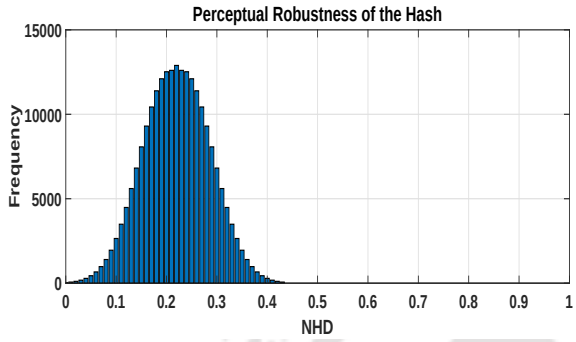
128-bit hashes were generated for 224 test sequences using the same secret key, and then, the NHD was measured between every possible pair of the hashes. The procedure was repeated for the remaining 999 secret keys. The histogram of the NHD values for this experiment is shown in Figure 4.6(b). The figure shows that the NHD measured between the hashes in this case approximately follows a Gaussian distribution with the mean centred



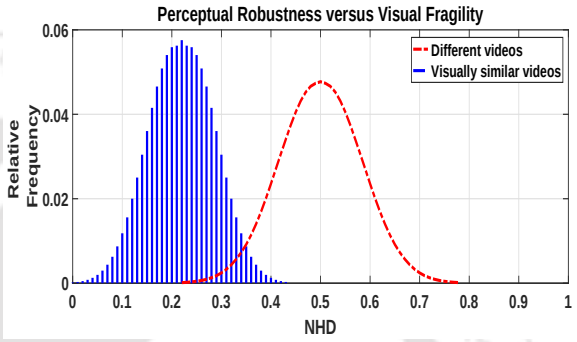
(a) Unpredictability property of the hash.



(b) Visual fragility property of the hash.



(c) Perceptual robustness property of the hash.



(d) Perceptual robustness property versus the visual fragility of the hash.

Figure 4.7: The histogram of normalized Hamming distance to evaluate the important desirable properties of the hash generated using the PARAFAC decomposition based video hashing algorithm.

around 0.5. A narrow-spread distribution centred around 0.5 implies that the perceptual hash closely satisfies the visually fragile property given in Equation 1.4. Comparing Figure 4.6(b) with Figure 3.27(b) we can conclude that the perceptual hash based on the Tucker decomposition satisfies the visual fragility property more closely than the perceptual hash based on the TWT-ARM.

(iii) *To validate the perceptual robustness property*

All the videos existing in the database were corrupted with AWGN of $\sigma = 81$ and subsequently blurred with the Gaussian mask of size 5×5 thereby generating a set of perceptually similar videos. The hashes were generated for these videos using the same secret key. The NHD between the hashes of the original videos and the perceptually similar videos was measured. A histogram skewed towards zero implies that the perceptual hash closely satisfies the perceptual robustness property given in Equation 1.3. The histogram of all the NHD values for this experiment is shown in the Figure 4.6(c) and we observe that the NHD between the original videos and the perceptually similar videos are distributed between 0 and approximately 0.4 with the mean centred around 0.2. Comparing Figure 4.6(c) with Figure 3.27(c) we conclude that the perceptual hash based on the Tucker decomposition closely satisfies the perceptual robustness property more closely than the perceptual hash based on the TWT-ARM.

Table 4.4: Comparison of hash validity among the PARAFAC decomposition and the Tucker decomposition based perceptual video hashing algorithms.

Name of the Video Hashing Algorithm	Unpredictability		Visual Fragility		Perceptual Robustness	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
PARAFAC decomposition	**0.5**	**0.076**	**0.5**	**0.164**	**0.215**	**0.127**
Tucker decomposition	**0.5**	**0.060**	**0.5**	**0.147**	**0.2**	**0.118**

(iv) *To assess perceptual robustness versus visual fragility*

To evaluate the results of perceptual similarity obtained from the proposed video hashing algorithm for the perceptually similar and perceptually distinct videos, the histograms of the NHD obtained for the perceptual robustness property and the visual fragility property are superimposed as shown in Figure 4.6(d). From the figure, it can be observed that the histogram of the NHD for the perceptually similar and distinct videos overlap. The overlapping indicates that there would be misclassification between the perceptually similar and distinct videos. This figure also shows that the overlap of these two superimpose histograms is quite less implying lower misclassification rates.

Further, the overlapping here is less compared to Figure 3.27(d). Hence, the performance of the video hashing algorithm based on the Tucker decomposition is better than the video hashing algorithm based on the TWT-ARM.

All the validation steps are repeated for the perceptual hash based on the PARAFAC decomposition and depicted in Figure 4.7.

The unpredictability, visual fragility and the perceptual robustness properties of the perceptual hashes generated using the Tucker decomposition and the PARAFAC decomposition are compared in terms of the mean and the standard deviation of the NHD in Table 4.4.

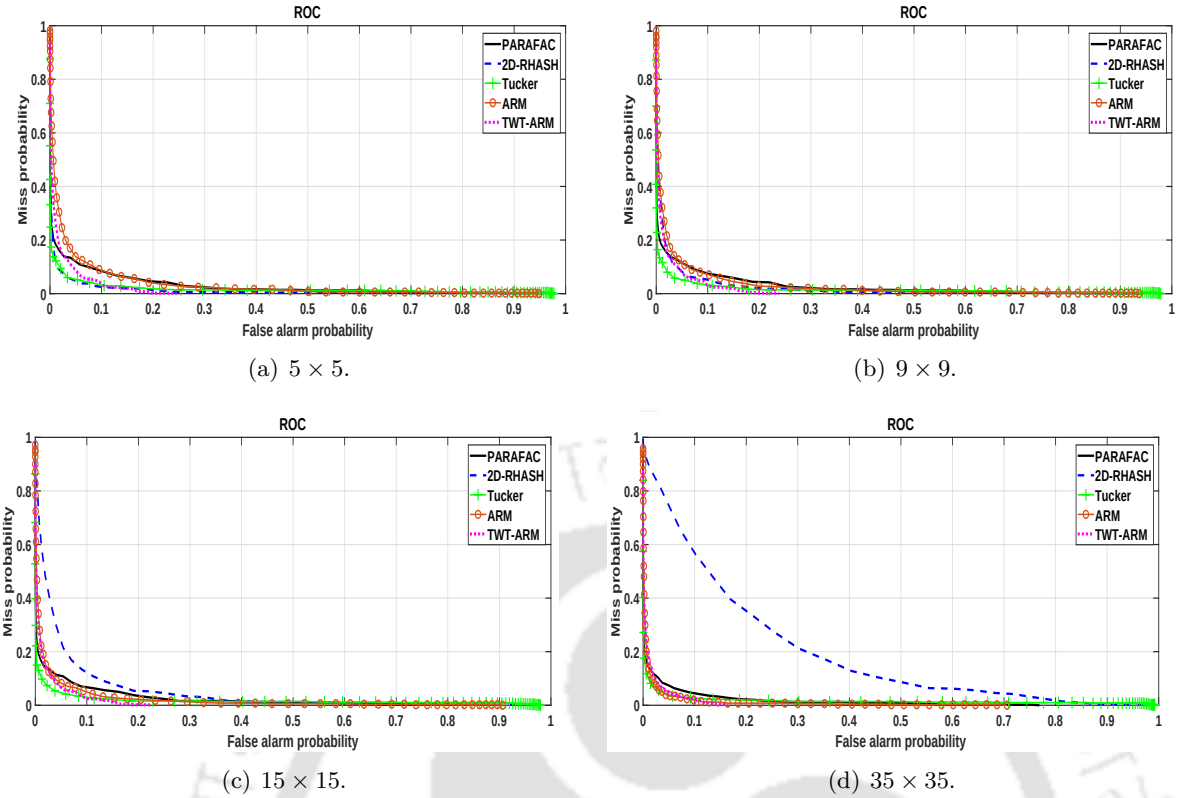
From Figure 4.6, Figure 4.7 and Table 4.4 we conclude that the perceptual hash generated using the Tucker decomposition satisfies all the desirable properties more efficiently.

4.4.3 Evaluation of Hash Performance

In this set of experiments, the performance of the Tucker decomposition based video hashing algorithm is compared with those of the video hashing algorithms based on the PARAFAC decomposition, the 2D-RASH, the ARM and the TWT-ARM. These algorithms showed competitive performance against most of the common image processing attacks and hence were chosen for comparison. The performance is evaluated in terms of the ROC curves [37]. The video hashing algorithms based on the Tucker decomposition, the PARAFAC decomposition, the 2D-RASH, the ARM and the TWT-ARM are labelled as *Tucker*, *PARAFAC*, *2D-RHASH*, *ARM* and *TWT-ARM* respectively. The same single/multiple image processing attacks as in Chapter 2 and Chapter 3 are considered here.

4.4.3.1 Single Attacks

The evaluation of hash performance for various single attacks is as follows. The attacks include the image processing and the malicious attacks described earlier.



(a) 5×5. (b) 9×9.

(c) 15×15. (d) 35×35.

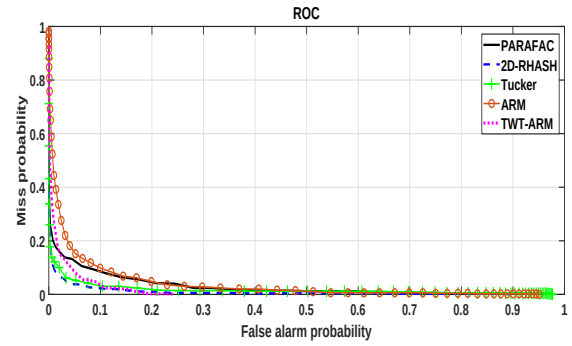
Figure 4.8: Statistical evaluation of video hashing algorithms via the ROC curves under average blurring attack using different mask size.

Average blurring attack: Figure 4.8 shows the ROC curves under average blurring using the mask of the following sizes: 5×5, 9×9, 15×15 and 35×35. The performance of the video hashing algorithms based on the Tucker decomposition, the PARAFAC decomposition, the ARM and the TWT-ARM is similarly good for the mean filter of all mask sizes considered. The performance of the video hashing algorithm based on the 2D-RASH deteriorates for the mask sizes 15×15 and 35×35.

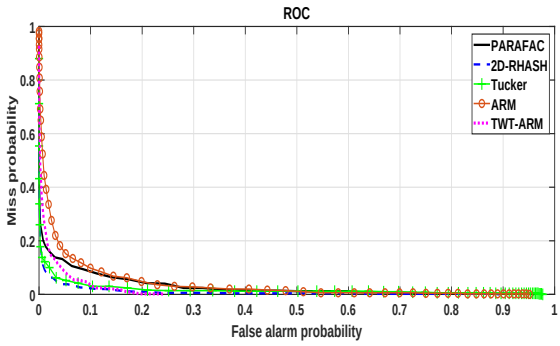
Gaussian blurring attack: The ROC curves for the Gaussian filtering using the 5×5, 9×9, 15×15 and 35×35 masks are shown in Figure 4.9. The performance of all the video hashing algorithms under considerations is similarly good for all the mask sizes. It can be observed that the performance of all these video hashing algorithms deteriorates as the mask size is increased to 15×15 and 35×35.

AWGN attack: Figure 4.10 shows the ROC curves for the AWGN attack for the following values of σ: 57, 81, 114 and 127. It is seen that the ARM-based video hashing algorithm is relatively stable followed by the hashing algorithms based on the Tucker decomposition, the PARAFAC decomposition and the TWT-ARM respectively. As σ increases, the performance of the video hashing algorithm based on the 2D-RASH deteriorates quickly.

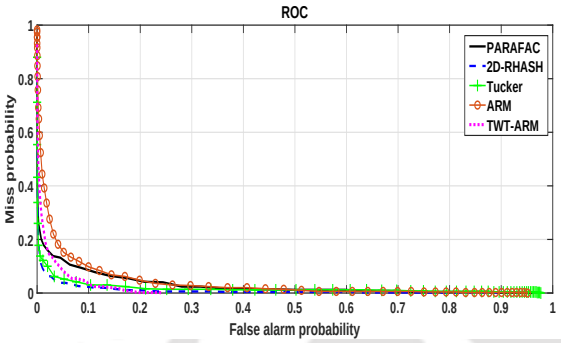
Salt and pepper noise attack: Figure 4.11 shows the ROC curves under the salt and pepper noise attack with the following noise densities: 5%, 10%, 20% and 25%. The ARM and the Tucker decomposition based video hashing algorithms are relatively stable followed by the PARAFAC decomposition and the TWT-ARM based video hashing algorithms. It can



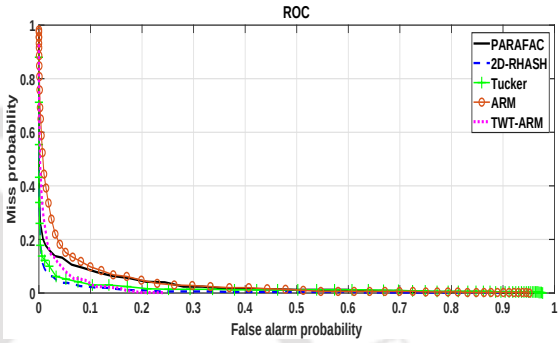
(a) 5×5.



(b) 9×9.

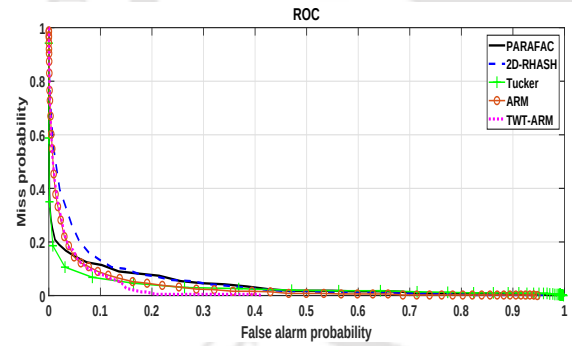


(c) 15×15.

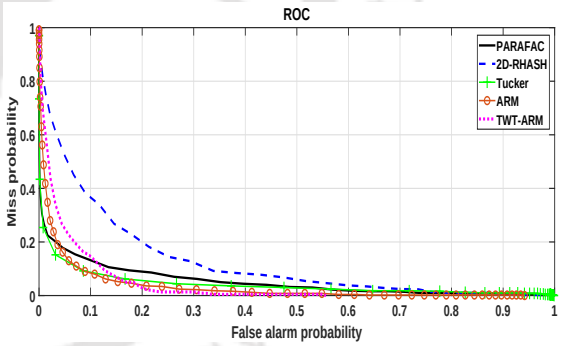


(d) 35×35.

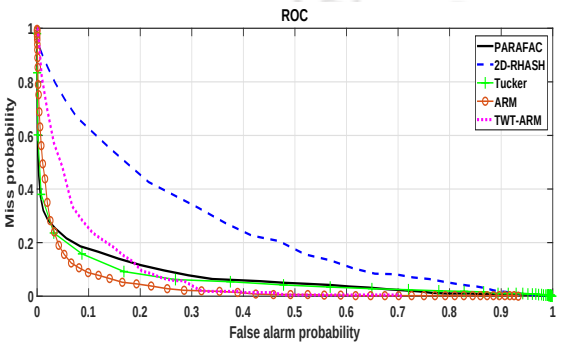
Figure 4.9: Statistical evaluation of video hashing algorithms via the ROC curves under Gaussian blurring attack using different mask size.



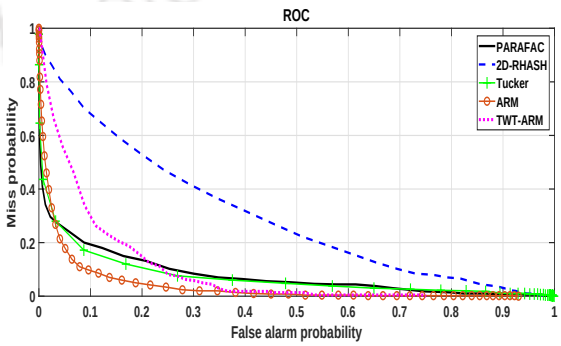
(a) Zero mean AWGN with $\sigma = 57$.



(b) Zero mean AWGN with $\sigma = 81$.

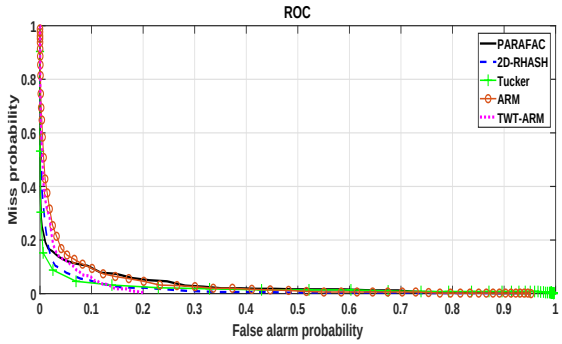


(c) Zero mean AWGN with $\sigma = 114$.

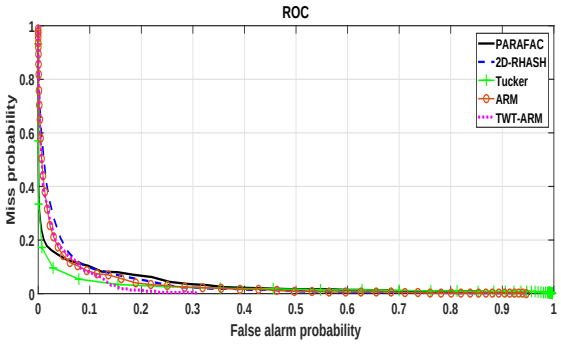


(d) Zero mean AWGN with $\sigma = 127$.

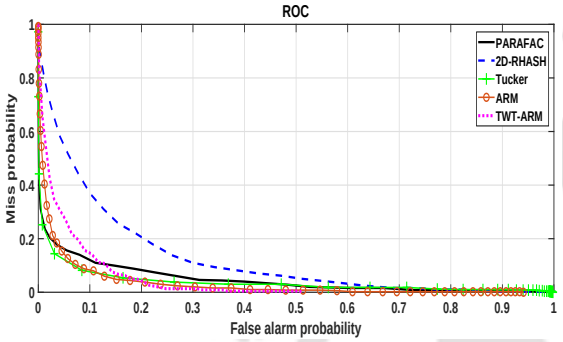
Figure 4.10: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN attack.



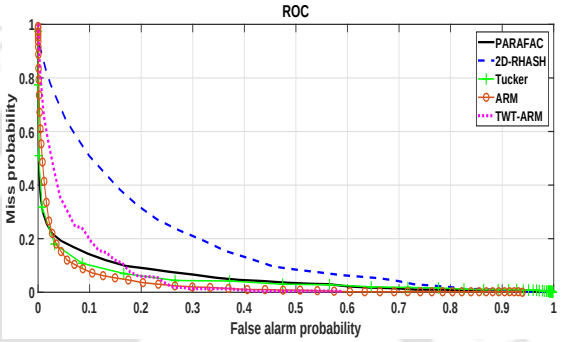
(a) Salt and pepper noise with 5% noise density.



(b) Salt and pepper noise with 10% noise density.



(c) Salt and pepper noise with 20% noise density.



(d) Salt and pepper noise with 25% noise density.

Figure 4.11: Statistical evaluation of video hashing algorithms via the ROC curves under salt and pepper noise attack.

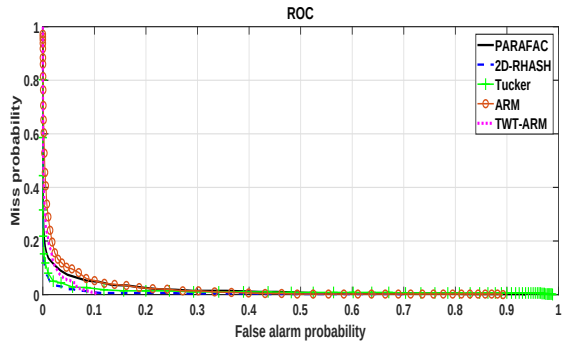
be observed that as the density of the noise increases, the performance of the video hashing algorithm based on the 2D-RASH degrades quickly.

Rotation attack: Figure 4.12 shows the ROC curves under the rotation attack with the following degrees of rotation: 2°, 5°, 8° and 12°. The performance of all the video hashing algorithms considered is similarly good for 2°, 5° and 8° rotation attack. For 12° rotation attack, the ARM-based video hashing algorithm has deteriorated.

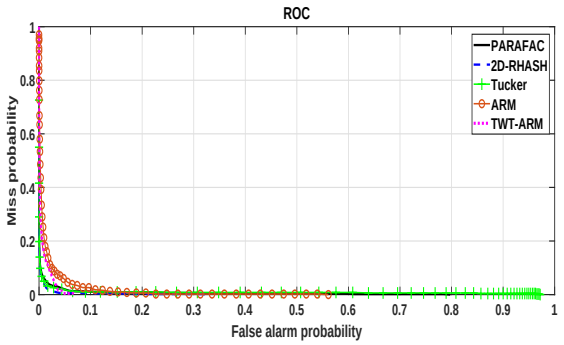
Cropping attack: The ROC curves for the cropping attack are shown in Figure 4.13. The following percentages of pixels were cropped from the borders of each frame: 5%, 10% and 15%. The performance of all the video hashing algorithms considered is similarly good except for the ARM-based video hashing algorithm. It can be observed from the figure that as more and more boundary pixels are cropped, the performance of the video hashing algorithm based on the ARM becomes progressively poor compared to the other video hashing algorithms.

Modified brightness attack: The ROC curves under the modified brightness attack (±5% and ±10%) are shown in Figure 4.14. The performance of the 2D-RASH based video hashing algorithm is relatively stable. The performance of all the other video hashing algorithms considered is more or less same except that of the TWT-ARM based video hashing algorithm showing poor performance.

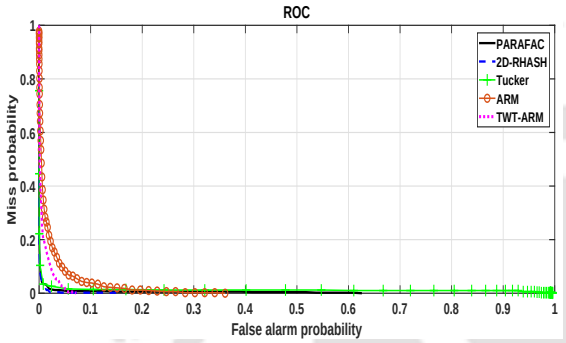
Modified contrast attack: The ROC curves for the modifications in the contrast are shown in Figure 4.15. The contrast of the frames are increased by multiplying the frame pixels with 1.1 and 1.2, and decreased by multiplying the frame pixels with 0.9 and 0.8, respectively.



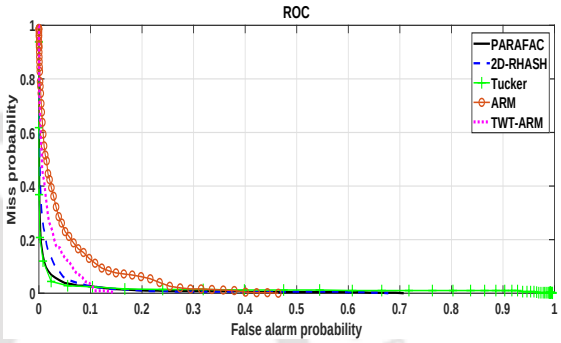
(a) 2°.



(b) 5°.

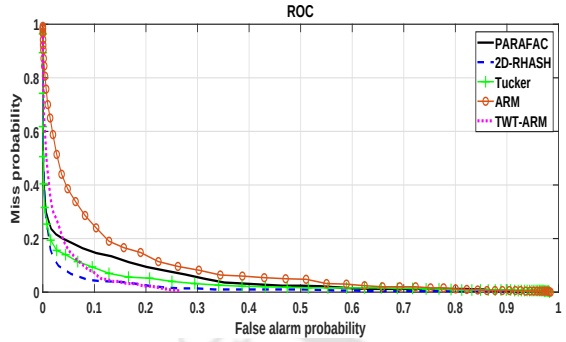


(c) 8°.

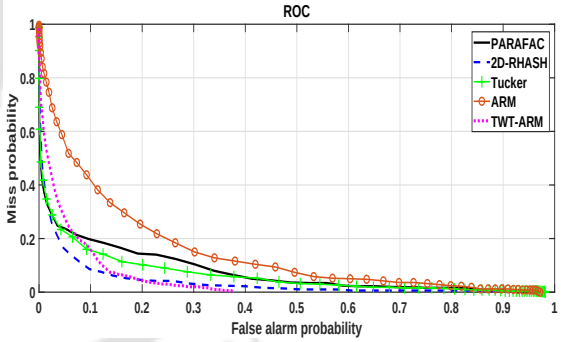


(d) 12°.

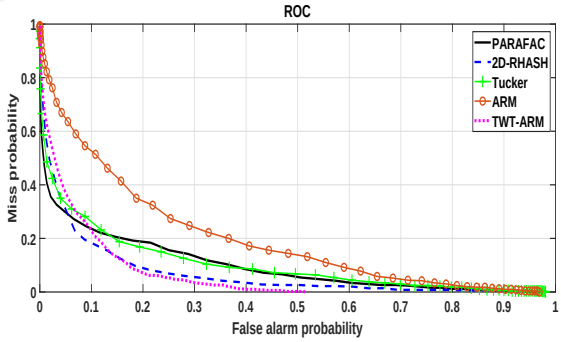
Figure 4.12: Statistical evaluation of video hashing algorithms via the ROC curves under rotation attack with different degrees of frame rotation.



(a) 5% crop.

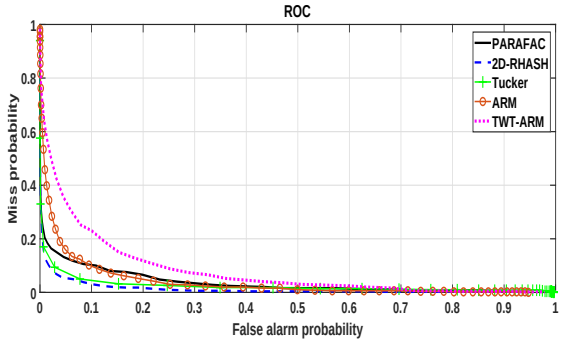


(b) 10% crop.

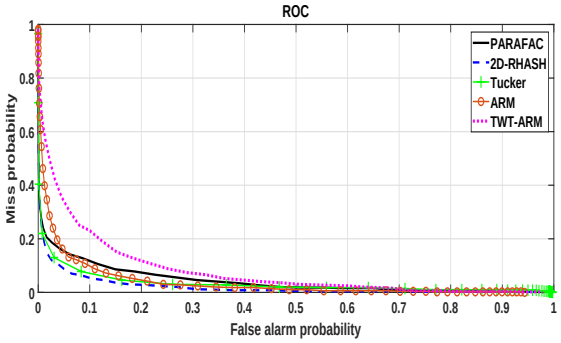


(c) 15% crop.

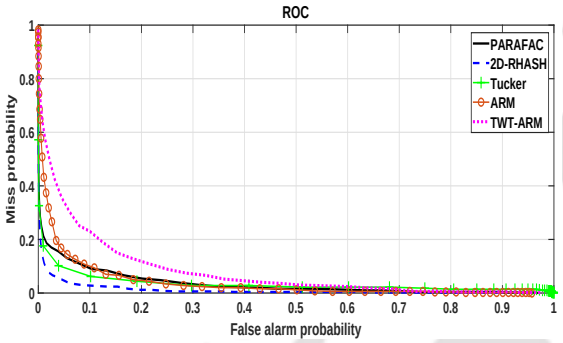
Figure 4.13: Statistical evaluation of video hashing algorithms via the ROC curves under cropping attack with different percentages of crop.



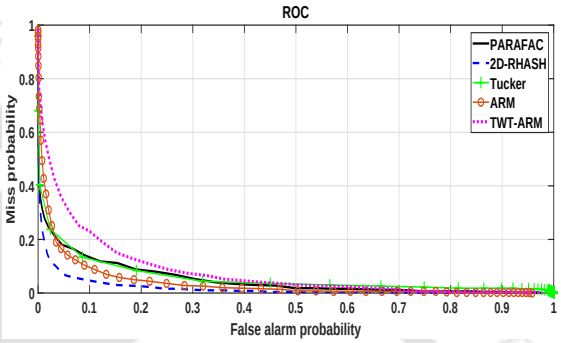
(a) 5% brightness increase.



(b) 10% brightness increase.

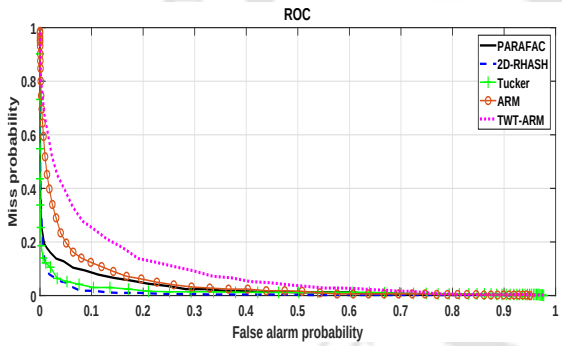


(c) 5% brightness decrease.

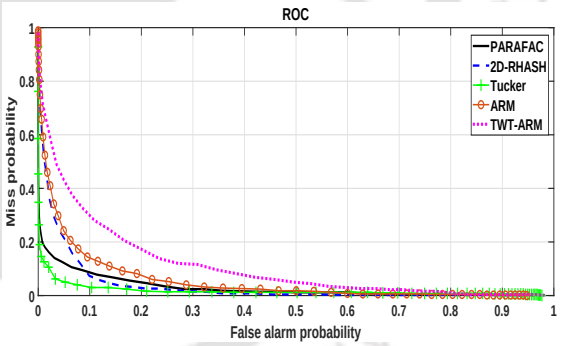


(d) 10% brightness decrease.

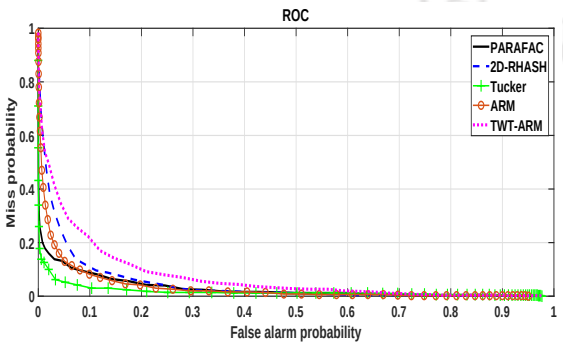
Figure 4.14: Statistical evaluation of video hashing algorithms via the ROC curves under modified brightness attack.



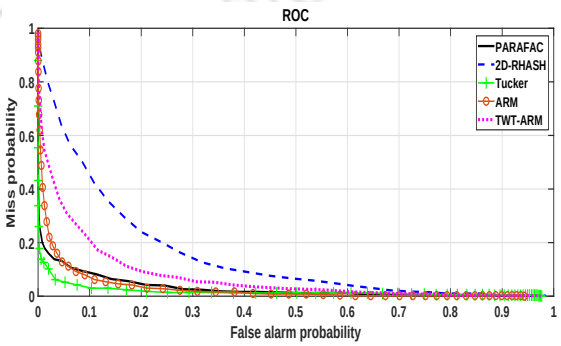
(a) 10% contrast increase.



(b) 20% contrast increase.

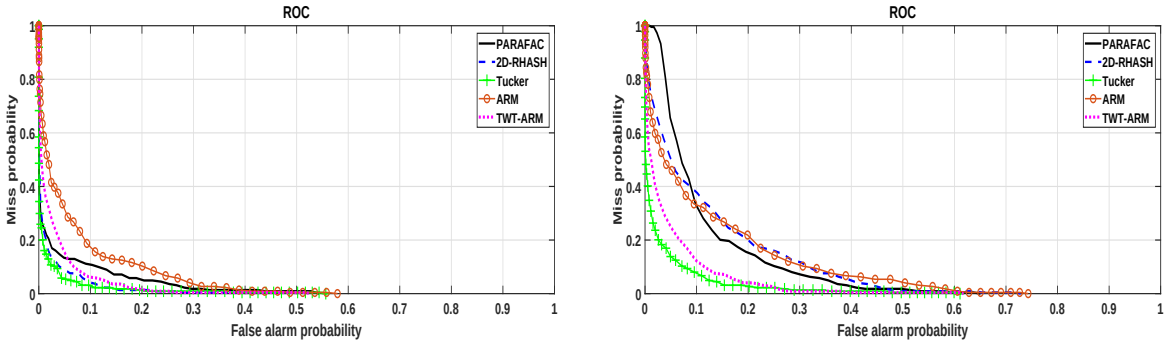


(c) 10% contrast decrease.



(d) 20% contrast decrease.

Figure 4.15: Statistical evaluation of video hashing algorithms via the ROC curves under modified contrast attack.



(a) 48 frames retained.

(b) 32 frames retained.

Figure 4.16: Statistical evaluation of video hashing algorithms via the ROC curves under frame-dropping and interpolation attack.

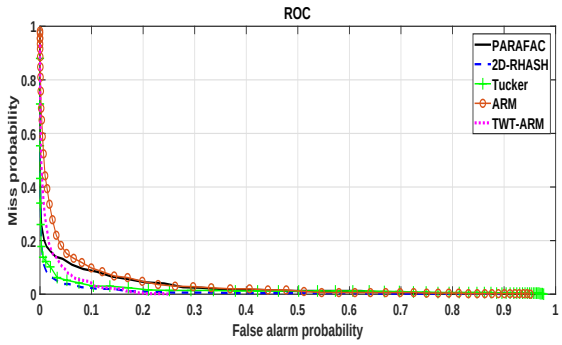
The video hashing algorithms based on the Tucker decomposition and the PARAFAC decomposition show robustness to contrast variations compared to the other video hashing algorithms considered. The TWT-ARM based video hashing algorithm is also relatively stable, but the area under the ROC curve is more. The 2D-RASH based video hashing algorithm is least robust to the contrast decreased attack.

Frame-dropping and interpolation attack: The ROC curves for the frame-dropping and interpolation attack are shown in Figure 4.16. In the first case, only 48 frames were retained, and then the remaining 16 frames were interpolated using temporal averaging. It can be observed that all the video hashing algorithms considered show similar performance except for the ARM-based video hashing showing relatively poor performance. In the second case, only 32 frames were retained, and then the remaining 32 frames were interpolated using temporal averaging. In this case, the Tucker decomposition based video hashing algorithm showed the maximum robustness to this attack followed by the video hashing algorithm based on the TWT-ARM. The remaining algorithms show relatively poor performance.

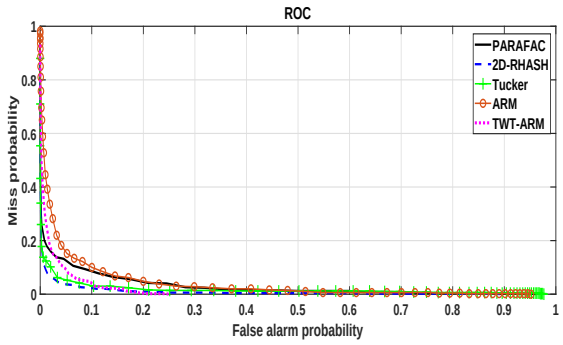
Spatial resolution variation attack: Figure 4.17 shows the ROC curves under the modified spatial resolutions attack. The following spatial resolutions were considered: 100×100, 250×250, 500×500 and 1000×1000. It can be observed that all the algorithms under considerations are having almost the same area under the ROC curves implying that the performance of all the video hashing algorithms considered is similar.

Compression attack: Two video database was constructed with the average CR of 250:1 and 240:1, respectively. The statistical evaluation of video hashing algorithms using the ROC curves under the compression attack is shown in Figure 4.18. It can be observed that for both the CRs, the performance of the video hashing algorithms based on the Tucker decomposition and the 2D-RASH has the least area under the ROC curves followed by the TWT-ARM, the PARAFAC decomposition and the ARM-based video hashing algorithms.

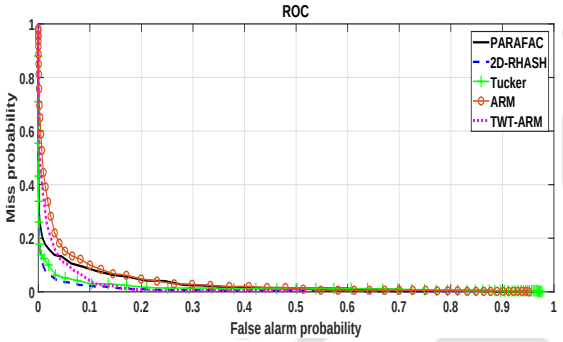
Reverse play attack: The ROC curves under the reverse play attack are shown in Figure 4.19. The TWT-ARM based video hashing algorithm was successful in having the least false alarm probability and the miss probability. The ROC curves for the remaining video hashing algorithms stayed close to the line of no discrimination implying the poor performance of all the algorithms considered.



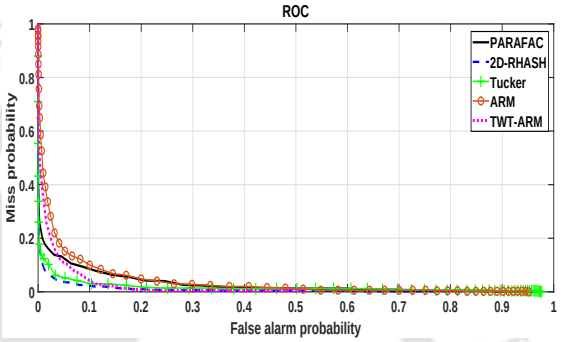
(a) 100×100.



(b) 250×250.

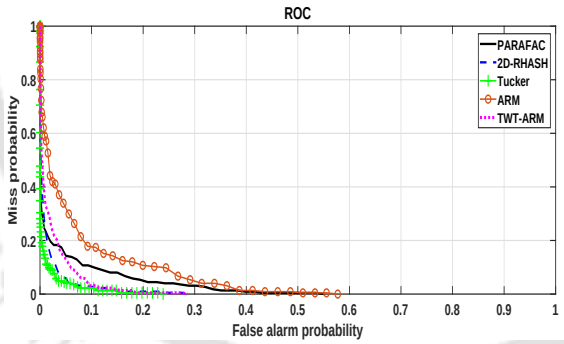


(c) 500×500.

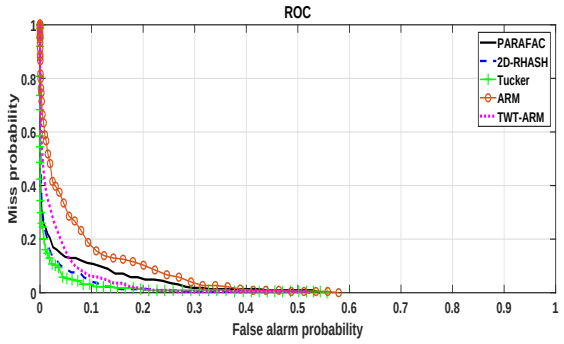


(d) 1000×1000.

Figure 4.17: Statistical evaluation of video hashing algorithms via the ROC curves under spatial resolution variation attack.



(a) 100kbps bit-rate and 240:1 average CR.



(b) 64kbps bit-rate and 250:1 average CR.

Figure 4.18: Statistical evaluation of video hashing algorithms via the ROC curves under compression attack.

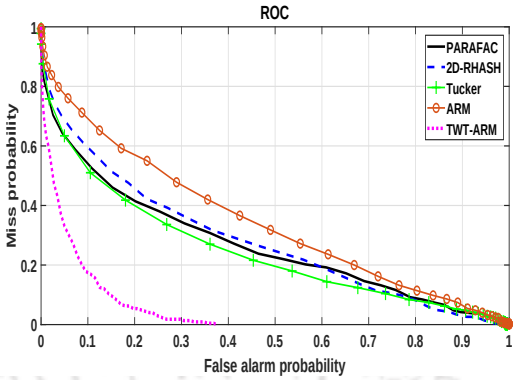
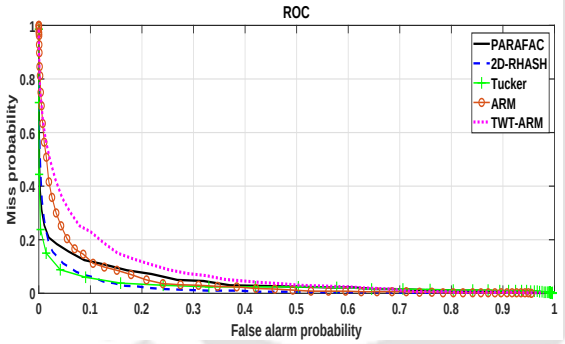
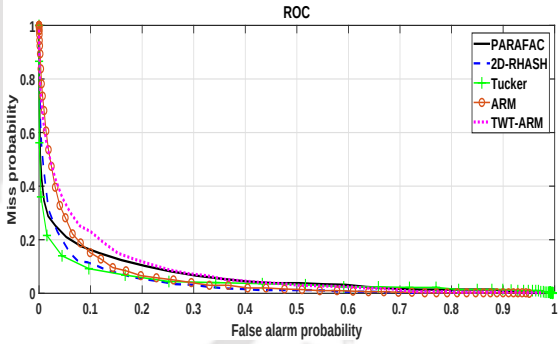


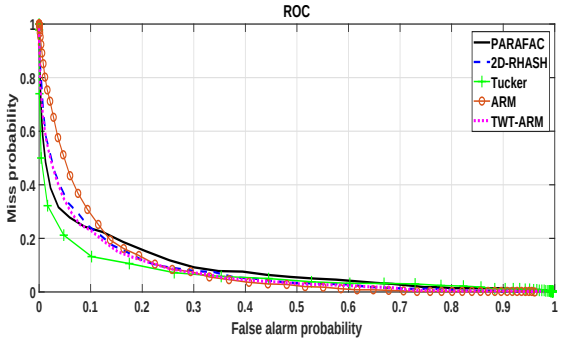
Figure 4.19: Statistical evaluation of video hashing algorithms via the ROC curves under reverse play attack.



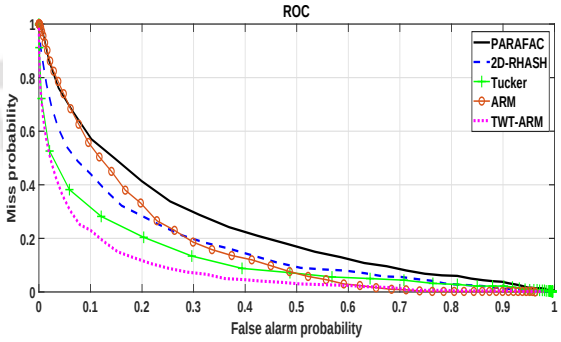
(a) 4 unwanted frames inserted.



(b) 8 unwanted frames inserted.

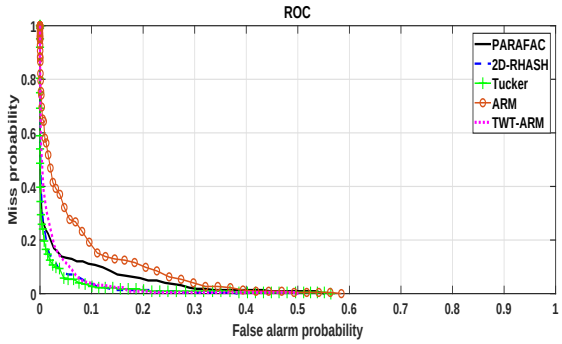


(c) 16 unwanted frames inserted.

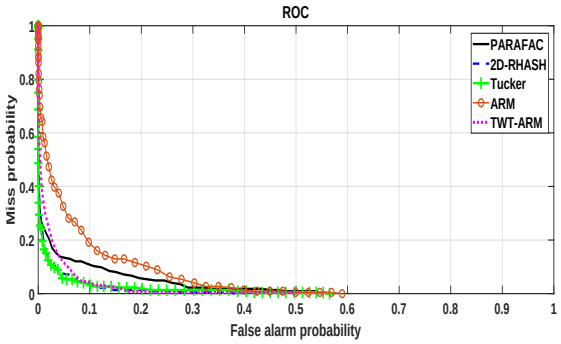


(d) 32 unwanted frames inserted.

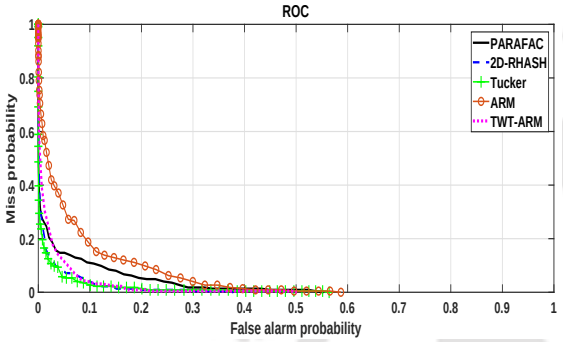
Figure 4.20: Statistical evaluation of video hashing algorithms via the ROC curves under frame insertion attack.



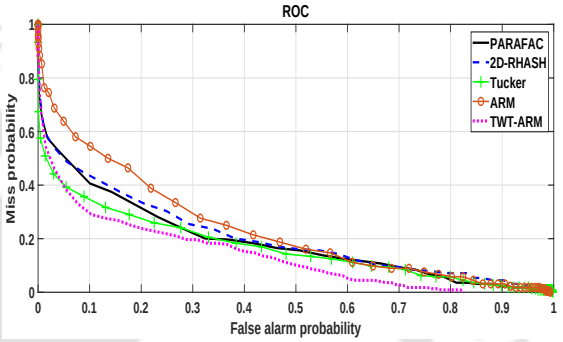
(a) Watermark of size 16×16.



(b) Watermark of size 32×32.



(c) Watermark of size 48×48.

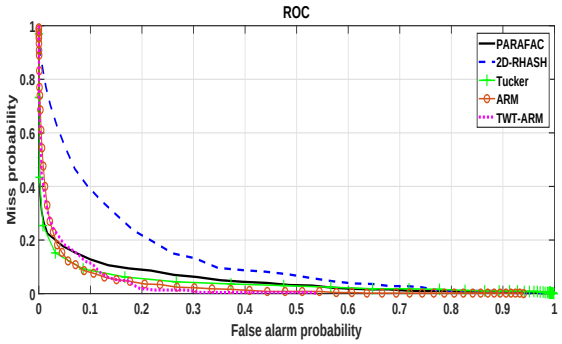


(d) Watermark of size 64×64.

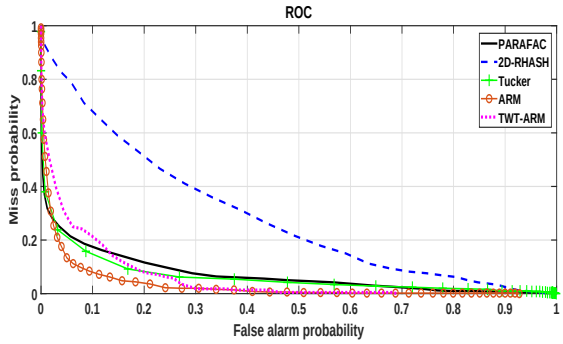
Figure 4.21: Statistical evaluation of video hashing algorithms via the ROC curves under watermark insertion attack.

Frame insertion attack: Firstly, four frames of a different video are inserted in the normalized version of the original video, and the performance of the algorithms are evaluated. Similarly, this attack is repeated by inserting eight, sixteen and thirty-two unwanted frames and then the performance of the algorithms are evaluated. The ROC curves under this attack are shown in Figure 4.20. This is a malicious attack, and hence the video hashing algorithm having the largest area under the ROC curve is said to be performing better. It can be observed that the TWT-ARM based video hashing algorithm is relatively stable in all the cases. When 4 or 8 different video frames are inserted, all the video hashing algorithm has failed to capture the malicious attack because of being perceptually robust. As the number of unwanted frame insertion increases, the PARAFAC decomposition based video hashing algorithm performs better followed by the video hashing algorithms based on the ARM, the 2D-RASH and the Tucker decomposition.

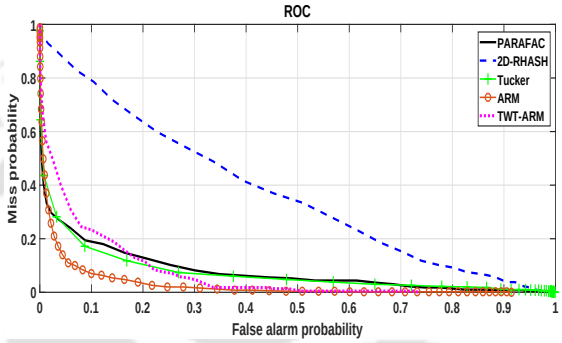
Watermark insertion attack: In this attack, a visible logo of different sizes is inserted to each video frame as a visible watermark. The following sizes of the logo are considered: 16×16, 32×32, 48×48 and 64×64. The ROC curves for this attack are shown in Figure 4.21. This is also considered to be a malicious attack, and hence the video hashing algorithm having the largest area under the ROC curve is considered to be performing better. For the size of the logo up to 48×48 the performance of the video hashing algorithms based on the Tucker decomposition and the 2D-RASH is having less area under the ROC curve, followed by the TWT-ARM, the PARAFAC decomposition, and the ARM-based video hashing algorithms



(a) Average blurring using a 3×3 mask and adding zero mean AWGN with $\sigma = 81$.



(b) Average blurring using a 5×5 mask and adding zero mean AWGN with $\sigma = 114$.



(c) Average blurring using a 9×9 mask and adding zero mean AWGN with $\sigma = 127$.

Figure 4.22: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN and average blurring attacks.

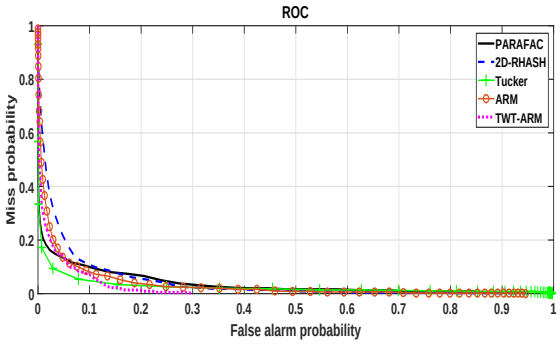
having relatively more area under the ROC curve. For the logo of size 64×64, the performance of all the video hashing algorithms considered is similarly good.

4.4.3.2 Multiple Attacks

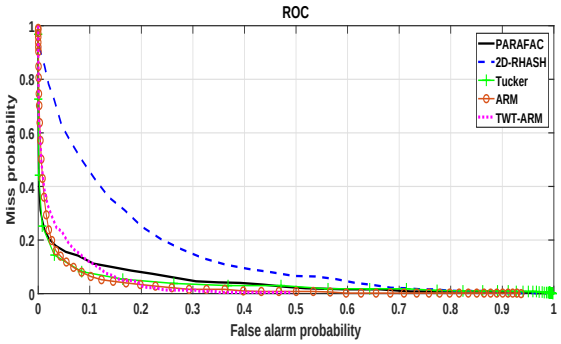
The evaluation of hash performance for various multiple attacks is as follows. The attacks include combination of various single image processing and malicious attacks described earlier.

AWGN and average blurring attacks: In this attack, each video frame was subjected to the average blurring with different mask sizes (3×3, 5×5 and 9×9), and then the addition of the zero mean AWGN with σ ranging from 81 to 127. The corresponding ROC curves are shown in Figure 4.22. For average blurring using a 3×3 mask and addition of zero mean AWGN with $\sigma = 81$, the performance of all the video hashing algorithms is good except for the 2D-RASH based video hashing algorithm. Similar observations are made when the video is subjected to average blurring using a 5×5 mask and zero mean AWGN with $\sigma = 114$ attacks. When the video is subjected to average blurring using a 9×9 mask and addition of zero mean AWGN with $\sigma = 127$, the video hashing algorithm based on the ARM was relatively stable, followed by the Tucker decomposition, the TWT-ARM and the PARAFAC decomposition based video hashing algorithms. The performance of the 2D-RASH based video hashing algorithm was poor under this attack.

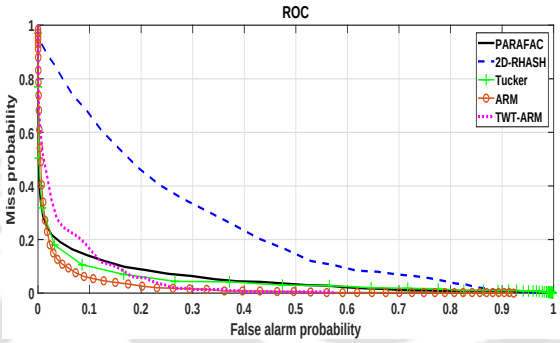
Salt and pepper noise and average blurring attacks: In this attack, each video frame



(a) Average blurring using a 3×3 mask and adding a salt and pepper noise with 5% noise density.



(b) Average blurring using a 5×5 mask and adding a salt and pepper noise with 20% noise density.

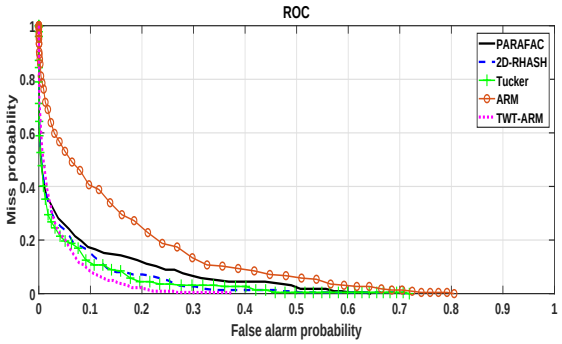


(c) Average blurring using a 9×9 mask and adding a salt and pepper noise with 25% noise density.

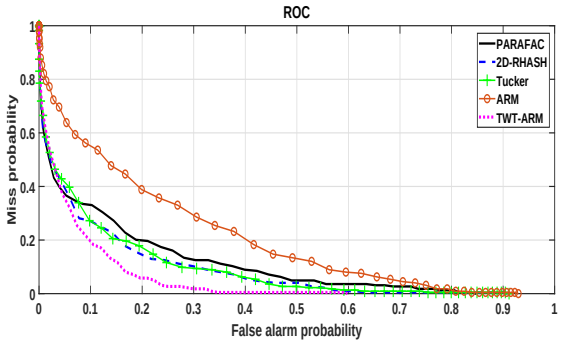
Figure 4.23: Statistical evaluation of video hashing algorithms via the ROC curves under salt and pepper noise and average blurring attacks.

was subjected to the average blurring with different mask sizes (3×3, 5×5 and 9×9), and then the addition of the salt and pepper noise with densities ranging from 5% to 25%. The corresponding ROC curves are shown in Figure 4.23. For average blurring using a 3×3 mask and addition of salt and pepper noise with 5% noise density, the performance of all the video hashing algorithms is good. For average blurring using a 5×5 mask and addition of salt and pepper noise with 20% noise density, the performance of all the video hashing algorithms is good except for the 2D-RASH based video hashing algorithm. Similar observations are made when the video is subjected to average blurring using a 9×9 mask and salt and pepper noise with 25% noise density attacks.

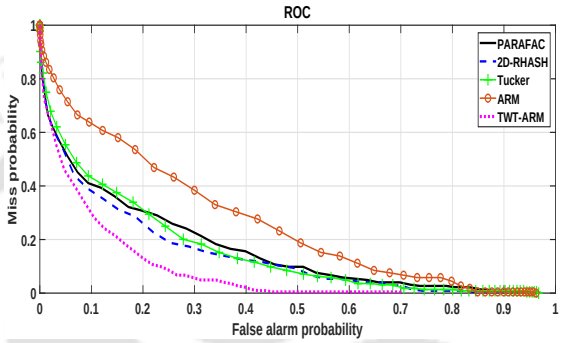
Cropping and Gaussian blurring attacks: In this attack, each video frame was subjected to the Gaussian filtering with different mask sizes (3×3, 5×5 and 9×9), and then the cropping of frame boundary pixels (5%, 10% and 15%). The corresponding ROC curves are shown in Figure 4.24. For the Gaussian blurring using a 3×3 mask and 5% cropping of frame boundary pixels, the performance of all the video hashing algorithms is good except for the ARM-based video hashing algorithm showing poor performance. For the Gaussian blurring using a 5×5 mask and 10% cropping of frame boundary pixels, the performance of the TWT-ARM based video hashing algorithm is relatively good followed by the Tucker decomposition, the PARAFAC decomposition and the 2D-RASH based video hashing algorithms showing similar performance. The ARM-based video hashing algorithm has poor performance in this case.



(a) Gaussian blurring using a 3×3 mask and 5% cropping.



(b) Gaussian blurring using a 5 × 5 mask and 10% cropping.



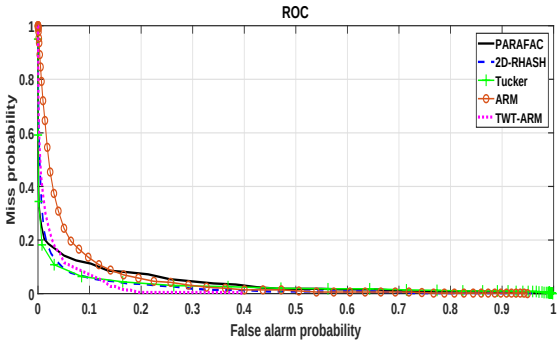
(c) Gaussian blurring using a 9 × 9 mask and 15% cropping.

Figure 4.24: Statistical evaluation of video hashing algorithms via the ROC curves under cropping and Gaussian blurring attacks.

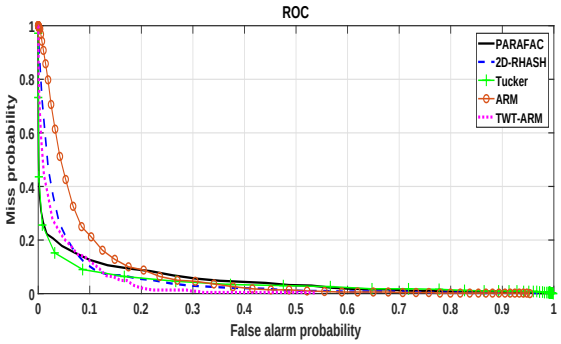
A similar observation is made for the Gaussian blurring using a 9 × 9 mask and 15% cropping of frame boundary pixels attacks.

AWGN and Gaussian blurring attacks: In this attack, each video frame was corrupted with zero mean AWGN with σ ranging from 57 to 114 and followed by the Gaussian filtering with different mask sizes (3×3, 5×5 and 9×9). The corresponding ROC curves are shown in Figure 4.25. For the addition of zero mean AWGN with $\sigma = 57$ and the Gaussian blurring using a 3×3 mask, the performance of all the video hashing algorithms is good. For the addition of zero mean AWGN with $\sigma = 81$ and the Gaussian blurring using a 5×5 mask, the performance of all the video hashing algorithms is good except for the ARM-based video hashing algorithm showing poor performance. Finally, when the video is subjected to the addition of zero mean AWGN with $\sigma = 114$ and the Gaussian blurring using a 9×9 mask, the performance of all the video hashing algorithms is good except for the ARM and the 2D-RASH based video hashing algorithms showing poor performance.

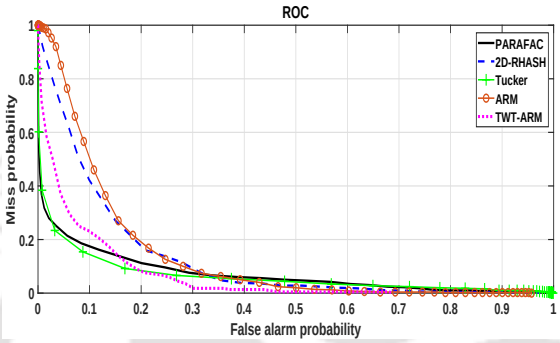
Frame-dropping and interpolation followed by rotation attacks: In this attack video frames are dropped at regular intervals while retaining only 16 or 8 of the original frames and then interpolated to obtain 64 number of frames, followed by 5° or 8° rotation attacks respectively. The ROC curves for these multiple attacks are shown in Figure 4.26. It can be observed that the video hashing algorithms based on the TWT-ARM and the Tucker decomposition have least area under the ROC curves indicating better performance. While the remaining



(a) Adding zero mean AWGN with $\sigma = 57$ and Gaussian blurring using a 3×3 mask.



(b) Adding zero mean AWGN with $\sigma = 81$ and Gaussian blurring using a 5×5 mask.



(c) Adding zero mean AWGN with $\sigma = 114$ and Gaussian blurring using a 9×9 mask.

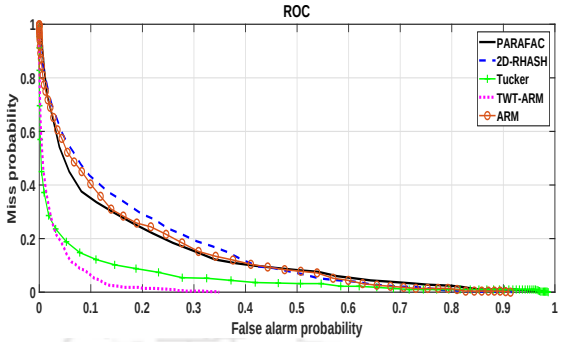
Figure 4.25: Statistical evaluation of video hashing algorithms via the ROC curves under AWGN and Gaussian blurring attacks.

video hashing algorithms considered, show relatively poor performance.

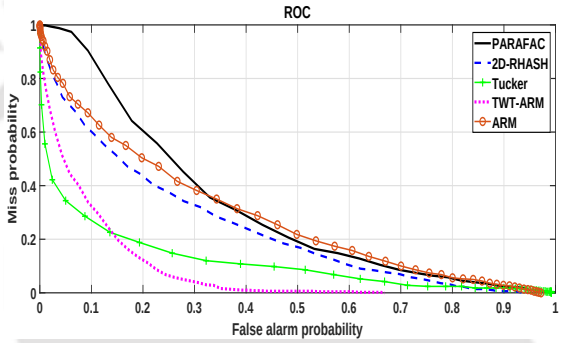
Increased spatial resolution, frame-dropping and interpolation followed by rotation attacks: In this attack, the spatial resolution of each frame was increased to 500×500 or 1000×1000, subjected to frame dropping at regular intervals while retaining only 32 or 16 of the original frames and then interpolated to obtain 64 number of frames, followed by 5° or 8° rotation attacks respectively. The ROC curves for these multiple attacks are shown in Figure 4.27. The area under the ROC curves is smaller for the video hashing algorithms based on the TWT-ARM and the Tucker decomposition. The other video hashing algorithms considered, show relatively poor performance.

Decreased spatial resolution, frame-dropping and interpolation followed by average blurring attacks: In this attack, the spatial resolution of each frame was decreased to 250×250 or 100×100, subjected to frame dropping at regular intervals while retaining only 32 or 16 of the original frames and then interpolated to obtain 64 number of frames, followed by mean blurring using a mask of 5×5 or 9×9 respectively. The ROC curves for these multiple attacks are shown in Figure 4.28. The area under the ROC curves is smaller for the video hashing algorithms based on the Tucker decomposition and the TWT-ARM. The other video hashing algorithms considered, show relatively poor performance.

Addition of AWGN, frame insertion and Gaussian blurring attacks: In this attack the video frames were subjected to zero mean AWGN with $\sigma = 81$ or $\sigma = 114$, insertion of 4 or

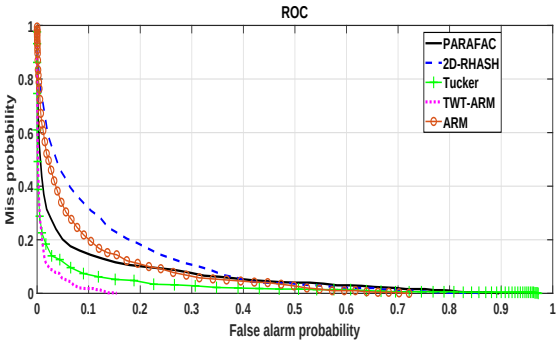


(a) Only 16 frames retained and 5° rotation.

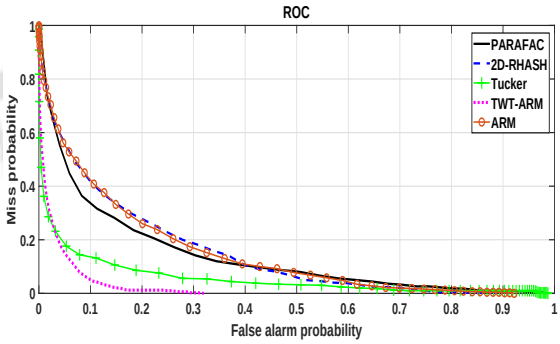


(b) Only 8 frames retained and 8° rotation.

Figure 4.26: Statistical evaluation of video hashing algorithms via the ROC curves under frame-dropping and interpolation followed by rotation attacks.

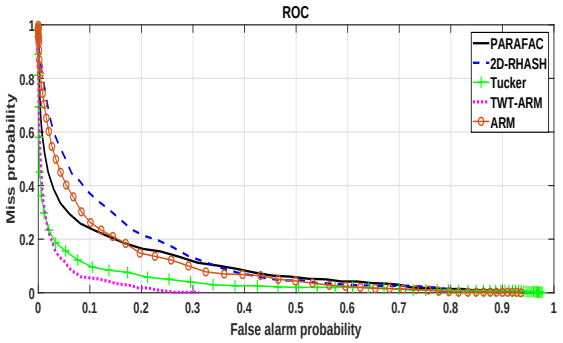


(a) Increase in spatial resolution to 500 × 500, only 32 frames retained and 5° rotation.

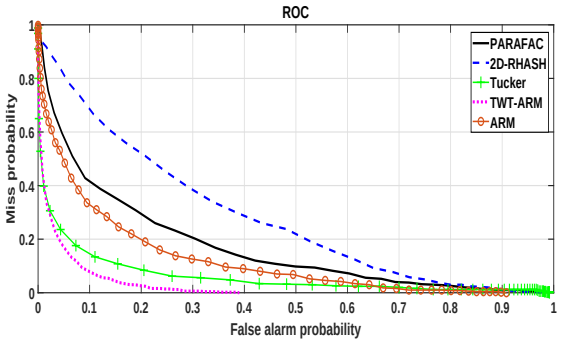


(b) Increase in spatial resolution to 1000 × 1000, only 16 frames retained and 8° rotation.

Figure 4.27: Statistical evaluation of video hashing algorithms via the ROC curves under increased spatial resolution, frame-dropping and interpolation followed by rotation attacks.

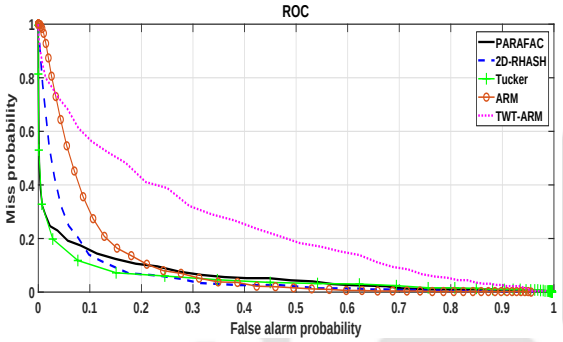


(a) Decrease in spatial resolution to 250×250, only 32 frames retained and average blurring using 5×5 mask.

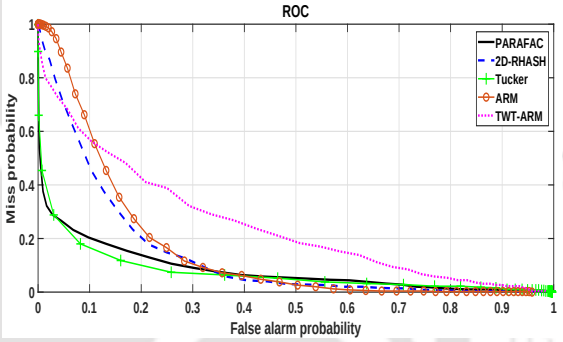


(b) Decrease in spatial resolution to 100×100, only 16 frames retained and average blurring using 9×9 mask.

Figure 4.28: Statistical evaluation of video hashing algorithms via the ROC curves under decreased spatial resolution, frame-dropping and interpolation followed by average blurring attacks.



(a) Adding a zero mean AWGN with $\sigma = 81$, 4 unwanted frames inserted and Gaussian blurring using a 5×5 mask.



(b) Adding a zero mean AWGN with $\sigma = 114$, 8 unwanted frames inserted and Gaussian blurring using a 9×9 mask.

Figure 4.29: Statistical evaluation of video hashing algorithms via the ROC curves under addition of AWGN, frame insertion and Gaussian blurring attacks.

8 unwanted frames and the Gaussian filtering with a mask size of 5×5 or 9×9 respectively. The corresponding ROC curves are shown in Figure 4.29. This is a combination of content-preserving attack and the malicious attack. The TWT-ARM based video hashing algorithm is having a relatively larger area under the ROC curves indicating better performance, followed by the video hashing algorithms based on the ARM and the 2D-RASH. The performance of the video hashing algorithms based on the Tucker decomposition and the PARAFAC decomposition are perceptually robust and fails to capture the malicious attack.

The performance of the video hashing algorithms based on PARAFAC decomposition, 2D-RHASH, 3D-RPT-2D-DCT, ARM, TWT-ARM and Tucker decomposition for various single and multiple attacks are tabulated in Table 4.5 and Table 4.6, respectively.

Table 4.5: Performance comparison of various video hashing algorithms under single attack

Attack	**PARAFAC**	**2D-RHASH**	**3D-RPT-2D-DCT**	**ARM**	**TWT-ARM**	**TUCKER**
Average blurring	Very good	Poor	Poor	Very good	Very good	Very good
Gaussian blurring	Very good	Very good	Very good	Very good	Very good	Very good
Zero mean AWGN	Good	Poor	Poor	Very good	Good	Good
Salt and pepper noise	Good	Poor	Poor	Good	Good	Good
Frame rotation	Very good	Very good	Very good	Very good	Very good	Very good
Cropping	Good	Good	Good	Good	Good	Good
Brightness modification	Very good	Very good	Good	Very good	Good	Very good
Contrast modification	Very good	Good	Poor	Very good	Good	Very good
Frame dropping and interpolation	Good	Good	Good	Good	Very good	Very good
Spatial resolution variation	Very good	Very good	Very good	Very good	Very good	Very good
Compression	Very good	Very good	Very good	Very good	Very good	Very good
Reverse play	Poor	Poor	Very good	Poor	Very good	Poor
Unwanted frame insertion	Very good	Good	Very good	Good	Good	Good
Watermark insertion	Good	Good	Good	Good	Good	Good

4.5 Computational Complexity

3D-DCT based Video Hashing Algorithm: In [13], [12], the video hash is obtained by applying the 3D DCT on the whole video data. The 3D DCT of a cube of $X \times X \times X$ elements are calculated in $\mathcal{O}\left(X^3 \log_2 X\right)$ operations.

Centroid of Gradient Orientations based Video Hashing Algorithm: In [7], [27], the CGO is used to obtain the hash from each frame of the video. Therefore, the computational complexity of the algorithm is X times the per-frame complexity. Further, each frame is divided into 4×2 blocks. The number of pixels in each block is $\frac{X}{4} \times \frac{X}{2} = \frac{X^2}{8}$ pixels. The CGO for each block is calculated using the equation,

$$c\left[n,m,k\right] = \frac{\sum\limits_{(x,y)\in B_{n,m,k}} r\left[x,y,k\right]\theta\left[x,y,k\right]}{\sum\limits_{(x,y)\in B_{n,m,k}} r\left[x,y,k\right]} \tag{4.14}$$

where, $B_{n,m,k}$ is the block in the n^{th} row and m^{th} column of the k^{th} frame, $c\left[n,m,k\right]$ is the centroid of the block $B_{n,m,k}$, $(1 \le n \le 2, 1 \le m \le 4)$

$$r\left[x,y,k\right] = \sqrt{G_x^2 + G_y^2} \tag{4.15}$$

Table 4.6: Performance comparison of various video hashing algorithms under multiple attack

Attack	**PARAFAC**	**2D-RHASH**	**3D-RPT-2D-DCT**	**ARM**	**TWT-ARM**	**TUCKER**
Zero mean AWGN and average blurring	Very good	Poor	Poor	Very good	Very good	Very good
Salt and pepper noise and average blurring	Very good	Poor	Poor	Very good	Very good	Very good
Cropping and Gaussian blurring	Very good	Good	Good	Poor	Very good	Good
AWGN and Gaussian blurring	Very good	Good	Poor	Good	Very good	Very good
Frame-dropping and interpolation followed by rotation	Good	Good	Good	Good	Very good	Very good
Increased spatial resolution, frame-dropping and interpolation followed by rotation	Good	Good	Good	Good	Very good	Very good
Decreased spatial resolution, frame-dropping and interpolation followed by average blurring	Very good	Poor	Poor	Very good	Very good	Very good
Addition of AWGN, frame insertion and Gaussian blurring	Very good	Good	Poor	Good	Good	Very good

is the gradient magnitude,

$$\theta\left[x,y,k\right]=\tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{4.16}$$

is the gradient direction,

$$G_x=f\left[x+1,y,k\right]-f\left[x-1,y,k\right] \tag{4.17}$$

is the approximation of gradient in x-direction,

$$G_y=f\left[x,y+1,k\right]-f\left[x,y-1,k\right] \tag{4.18}$$

is the approximation of gradient in y-direction, and $f\left(x,y,k\right)$ is the luminance value at location (x,y) in the k^{th} frame. Equation 4.15 includes two multiplications, Equation 4.16 includes one multiplication and Equation 4.14 (in the numerator) includes one multiplication, resulting in 4 multiplications per pixel. In addition, Equation 4.14 includes one multiplication per block.

Therefore, the computational complexity of every block is $\mathcal{O}\left(\frac{X^2}{8} \times 4 + 1\right) = \mathcal{O}\left(\frac{X^2}{2} + 1\right)$. Hence, the total number of operations include $\mathcal{O}\left(X \times 2 \times 4 \times \left(\frac{X^2}{2} + 1\right)\right) = \mathcal{O}\left(4X^3 + 8X\right)$.

Radial Projection based Video Hashing Algorithm: In [10], [11], the video hash is obtained by using radial projection of luminance values and 1D DCT in each frame. Thus, the computational complexity of the algorithm is X times the per-frame complexity. The radial projection is calculated using,

$$P(\phi) = \frac{\sum\limits_{(x,y)\in\Gamma(\phi)} I^2(x,y)}{\#\Gamma(\phi)} - \left(\frac{\sum\limits_{(x,y)\in\Gamma(\phi)} I(x,y)}{\#\Gamma(\phi)}\right)^2 \tag{4.19}$$

where $\Gamma(\phi)$ denote the set of pixels (x, y) that are located on the projection line corresponding to a given angle ϕ, $0° \leq \phi \leq 180°$ and $I(x, y)$ denote the luminance value of pixel (x, y). Each projection line has X number of pixels and there are such 180 radial projection with equal space. Equation 4.19 includes one multiplication per pixel and three multiplications for each projection line. The computational complexity for 180 projection lines is $180 \times (\mathcal{O}(X) + 4) = \mathcal{O}(180X + 720)$. Lastly, 1D DCT is applied on normalized projected vector resulting in $\mathcal{O}(X\log_2 X)$ operations. Finally, the total number of computations required to generate the video hash is $X \times (\mathcal{O}(180X + 720) + \mathcal{O}(X\log_2 X)) = \mathcal{O}\left(180X^2 + 720X + X^2\log_2 X\right)$.

PARAFAC Decomposition based Video Hashing Algorithm: In [14], [15], the random sub-videos extracted from a video, are modelled as third-order tensors. Using the PARAFAC tensor approximations, each sub-video is approximated using a rank one tensor. Therefore, the computational complexity of the algorithm depends on the number of operations required to find a rank-one tensor for a given sub-video. In this regard, let the dimension of each sub-video be denoted by $U \times U \times U$, where $U < \frac{2}{3}X$. From [74], it is found that the operations required to obtain a rank-one tensor for a sub-video of size $U \times U \times U$ is $\mathcal{O}\left(U^3\right)$. The authors in [14] and [15] have chosen eight sub-videos of size $U \times U \times U$ from a given video, to generate a perceptual hash. Therefore, the total number of computations required to generate a video hash is $8 \times \mathcal{O}\left(U^3\right) = 8 \times \mathcal{O}\left(\left(\frac{2}{3}X\right)^3\right) = \mathcal{O}\left(\frac{64}{27}X^3\right)$.

Scheme-I of the 3D-RASH based Video Hashing Algorithm: The computational complexity of three-dimensional radial projection is a straight forward extension of two-dimensional radial projection. In Scheme-I, the 3D-RASH are applied on N sub-cubes of dimension $U \times U \times U$, where $U = \frac{X}{4}$ are extracted from a video. N 3D-RASH vectors are obtained by applying 180×180 projection on each of the sub-cube. $\mathbf{r}_\mu$ is obtained by applying averaging operation on N 3D-RASH vectors and then the 1D-DCT is applied on $\mathbf{r}_\mu$. The first k DCT coefficients are chosen to obtain the intermediate hash vector. Thus, the computational complexity of the proposed Scheme-I of the 3D-RASH is,

$$\begin{aligned} &N \times [180 \times 180 \times \{\mathcal{O}(U) + 4\}] + 1 + \mathcal{O}(U\log_2 U) \\ &= N \times \left[32,400 \times \left\{\mathcal{O}\left(\tfrac{X}{4}\right) + 4\right\}\right] + 1 + \mathcal{O}\left(\tfrac{X}{4}\log_2\left(\tfrac{X}{4}\right)\right) \end{aligned}$$

Scheme-II of the 3D-RASH based Video Hashing Algorithm: In Scheme-II of the proposed 3D-RASH, the N array of variances obtained from 180×180 3D-RASH on each of the sub-cube is matricized to form $\mathbf{R}_{mat_i}$, $i = 1, 2, \ldots, N$. The 2D-DCT is applied on each $\mathbf{R}_{mat_i}$s,

and then the transformed matrix is averaged. Low-to-mid frequency coefficients are selected to obtain the intermediate hash vector. Therefore, the computational complexity of the proposed Scheme-II of the 3D-RASH is,

$$N \times [180 \times 180 \times \{\mathcal{O}(U) + 4\}] + N \times \mathcal{O}\left(U^2 \log_2 U\right) + 1$$
$$= N \times \left[32,400 \times \left\{\mathcal{O}\left(\tfrac{X}{4}\right) + 4\right\}\right] + N \times \mathcal{O}\left(\tfrac{X^2}{16} \log_2\left(\tfrac{X}{4}\right)\right) + 1$$

Achlioptas's random projection based Video Hashing Algorithm: The hash generated using the ARM-based video hashing algorithm involves only addition and subtractions due to the ARM matrix with elements 1 and -1. The multiplication is done only during the normalization after the addition/subtraction is completed. This multiplication is independent of the number of data points. Since, the complexity for all the video hashing algorithms is considered only in terms of the number of multiplications and not the number of additions required to generate the final hash, the computational complexity for the ARM-based video hashing algorithms is given by $\mathcal{O}(\zeta)$, where ζ is a constant. For the generation of the ARM-based hash from the video, the multiplication was performed only once during normalization. Therefore, $\zeta = 1$.

The TWT and the Achlioptas's random projection based Video Hashing Algorithm: In the TWT-ARM based video hashing algorithm, the videos are spatially normalized to a size of $X \times X$ without applying the normalization in the temporal direction. The TWT is applied in the temporal direction using the Haar as the mother wavelet. The low-pass frames of the wavelet decomposition of the video frames in the temporal direction capture the important visual features of the sequence. Considering only the low pass frames reduces the data and hence the computational complexity in the further steps. Furthermore, for a fixed-length hash, a fixed number of low-pass frames are required and hence the decomposition beyond a level is not considered. The computational complexity of the TWT for first level of wavelet decomposition with NoF number of frames in the video is given by $\mathcal{O}(NoF \times X \times X)$. Similarly, the computational complexity of the TWT for the second, third, fourth level decomposition and so on are given by $\mathcal{O}\left(\frac{NoF}{2} \times X \times X\right)$, $\mathcal{O}\left(\frac{NoF}{4} \times X \times X\right)$, $\mathcal{O}\left(\frac{NoF}{8} \times X \times X\right)$ and so on, respectively. In the video database considered, the maximum number of frames present in a video is 11,415. The temporal averaged frames are inserted to make this number equal to the power of two is 16,384. To extract 64 low pass frames using the TWT, eight levels of decomposition are required. N sub-cubes of dimension $U \times U \times U$, where $U = \frac{X}{4}$ are selected randomly from the transformed data of dimension $X \times X \times X$. The ARM which involves only addition and subtraction is applied to reduce the dimensionality and thus, the computational complexity of the ARM is given by $\mathcal{O}(\zeta)$, where ζ is a constant. Thus, the total number of operations involved to generate video hash using the proposed algorithm is $\mathcal{O}(NoF \times X \times X) + \mathcal{O}\left(\frac{NoF}{2} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{4} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{8} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{16} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{32} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{64} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{128} \times X \times X\right) + \mathcal{O}(\zeta)$ operations.

Tucker decomposition based Video Hashing Algorithm: In the Tucker decomposition based video hashing algorithm, N random sub-videos of dimension $U \times U \times U$, where $U = \frac{X}{2}$ are extracted from a video and modelled as third-order tensors. These tensors are decomposed

into $U \times 1$, $U \times 2$ and $U \times 1$ component matrices and core tensor with dimension $1 \times 2 \times 1$ using the method of the Tucker decomposition. From [75], the computational complexity of the Tucker decomposition algorithm for one tensor is given by $\mathcal{O}(U \times 1 \times 2 \times 1)$. Therefore, for N tensors (authors have chosen $N = 7$) the total number of computations required to generate a video hash is $N \times \mathcal{O}(U \times 1 \times 2 \times 1) = 7 \times \mathcal{O}\left(\left(\frac{X}{2}\right) \times 2\right) = \mathcal{O}(7X)$.

The computational complexity of all the video hashing algorithms under consideration is tabulated in Table 4.7. From the above discussions and Table 4.7, it can be concluded that the video hashing based on the ARM has the least computational complexity followed by the video hashing based on the Tucker decomposition.

Table 4.7: Computational complexity of all the perceptual video hashing algorithms under consideration.

Name of the perceptual video hashing algorithm (PVHA)	**Computational complexity**
3D-DCT based PVHA	$\mathcal{O}\left(X^3 \log_2 X\right)$
CGO based PVHA	$\mathcal{O}\left(4X^3 + 8X\right)$
2D-RASH based PVHA	$\mathcal{O}\left(180X^2 + 720X + X^2 \log_2 X\right)$
PARAFAC decomposition based PVHA	$\mathcal{O}\left(\frac{64}{27}X^3\right)$
Scheme-I of the 3D-RASH based PVHA	$N \times \left[32,400 \times \left\{\mathcal{O}\left(\frac{X}{4}\right) + 4\right\}\right] + 1 + \mathcal{O}\left(\frac{X}{4}\log_2\left(\frac{X}{4}\right)\right)$
Scheme-II of the 3D-RASH based PVHA	$N \times \left[32,400 \times \left\{\mathcal{O}\left(\frac{X}{4}\right) + 4\right\}\right] + N \times \mathcal{O}\left(\frac{X^2}{16}\log_2\left(\frac{X}{4}\right)\right) + 1$
Achlioptas's random projection based PVHA	$\mathcal{O}(\zeta)$
The TWT and the Achlioptas's random projection based PVHA	$\mathcal{O}(NoF \times X \times X) + \mathcal{O}\left(\frac{NoF}{2} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{4} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{8} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{16} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{32} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{64} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{128} \times X \times X\right) + \mathcal{O}(\zeta)$
The Tucker decomposition based PVHA	$\mathcal{O}(7X)$

4.6 Concluding Remarks

The perceptual hash for the original video (akiyo_cif), videos subjected to various attacks and the corresponding NHD using the different video hashing algorithms are shown in Table 4.8. The hash is represented using Hexadecimal numbers. The hash variations for the TWT-ARM and the Tucker decomposition based video hashing algorithms are less whereas the other hashing algorithms show substantial variations.

The performance of the the Tucker decomposition based video hashing algorithm was found to be robust to the following single image processing attacks: average blurring, Gaussian blurring, addition of AWGN, salt and pepper noise, rotation, cropping, modified brightness, modified contrast, frame dropping and interpolation, modified spatial resolutions, compression and insertion of a logo as the watermark. For most of the multiple attacks, the performance of the proposed video hashing algorithm was found to be superior than the well performing video hashing algorithms considered.

Table 4.8: The perceptual hash and the NHD between the original video and different distorted versions.

akiyo_cif	**PARAFAC**	**3D-RASH Scheme 1**	**TWT-ARM**	**Tucker decomposition**
Original Hash	000003FFFFF00FF0 07FC03FE1F8007FC	000BF8DD9962BEAE 304796F2033F70B7	D2244CA290FF7A03 DA738A4C3ECFD790	7FFFFFFF000000004 0000000FFFFFFFF
5° rotation	000001FFFFF05FF0 07FE03FC0F8007FC	0F0DE9FD9902FE2E 304996F2833C7075	D2244CA2D0FF7A03 DA728A4C3ECFD790	7FFFFFFF000000004 0000000FFFFFFFF
NHD	0.1563	0.4063	0.0625	0
Gaussian blurring (5×5)	000001FFFFF01FF0 07FF03FC0F8007FC	0E88F9FD9962BEA6 722A9652033DF035	D2244CA290FF7A03 DA738A4C3ECFD790	7FFFFFFF000000004 0000000FFFFFFFF
NHD	0.1563	0.5	0	0
Average blurring ($5\times$ 5)	000001FFFFF01FF0 07FF03FC0F8007FC	0E8CF9FD9902BEA6 722A9672033DF035	D2244CA290FF7A03 DA738A4C3ECFD790	7FFFFFFF000000004 0000000FFFFFFFF
NHD	0.1563	0.4688	0	0
AWGN ($\mu = 0, \sigma = 57$)	000001FFFFF01FF0 07FF03FE0F8007F8	0E8CF9DD9942BEE 6322A9672033DF035	D2244CA290FF7A03 DA738A4C3ECFD790	DFFFFFFF000000004 0000000FFFFFFFF
NHD	0.1563	0.4688	0	0.0313
10% Cropping	00000FFFFFF003FE0 003FFFE0B8007FC	8D6DEADF9942BC2 6304196D2C37D70A5	D22C4CA290FF7A03 DA728A1C3ECFD790	7FFFFFFF000000004 0000000FFFFFFFF
NHD	0.3125	0.5625	0.0938	0
10% Brightness increase	000001FFFFF03FF0 07FF03FC0B8007FC	0E88F9FD9962BEA 6722A9652033DF035	D2244CA290FF7A03 DA738A4C3ECFD790	DFFFFFFF000000004 0000000FFFFFFFF
NHD	0.1875	0.4688	0	0.0313

From the validation and the performance evaluation of the hash, it can be concluded that the performance of the Tucker based video hashing algorithm is better than the TWT-ARM based video hashing algorithm for most of the typical image processing attacks. Furthermore, the Tucker decomposition based video hashing algorithm has an added advantage of being computationally efficient.

4.7 Summary of the Chapter

This chapter is summarized as follows.

- The focus of this work was to treat the videos as tensors and extend the success of subspace projection of tensors based video hash generation such as PARAFAC decomposition.
- A generalized Tucker decomposition method was proposed for the generation of perceptual hashes from the videos.
- It covers the PARAFAC model as a special case and has been applied for the decomposition and interpretation of multi-way data arrays in many applications. Being a generalized tool, this model also allows the users to select a different number of factors along each mode, during the multi-way data array decomposition, aiding better analysis.
- A method was proposed for choosing a suitable number of components in the factor matrices of the Tucker decomposition.

- The hash was validated for the desirable properties of the perceptual hash, and it was found to be very satisfactory.
- The performance of the proposed method was evaluated using the ROC curves and compared with video hashing based on the ARM, the 2D- RHASH, the PARAFAC decomposition and the TWT-ARM.
- The algorithm showed excellent robustness to most of the single and multiple image processing attacks except the malicious attack.
- Comparison of all the proposed video hashing algorithms and the video hashing algorithms under consideration were performed in terms of the computational complexity.

Chapter 5

Application of Perceptual Video Hashing for Near-duplicate Video Retrieval

Contents of the Chapter

The rapid pace of growth in the VLSI technology and the Internet has led to an enormous increase in the number of videos generated or viewed. Furthermore, the easy availability of a number of video editing tools such as *Freemake*, *Blender*, *Shotcut*, *Lightworks* has also led to the significant increase in percentage of *near-duplicate videos* (NDVs) in the online databases. According to Susan Wojcicki, the CEO of the YouTube, '400 hours of video are uploaded on the YouTube by users every minute as on July 2015' [76]. The users may maliciously modify the downloaded YouTube video and upload it back to the YouTube database server, violating the copyrights and also increasing the amount of redundancy and adding burden on the server. Thus, there is a strong need for the perceptual content of the video to be checked by the system before granting permission to the user for uploading the video. This leads to video copyright protection and also an efficient video database management by disallowing the user to upload the video if perceptually similar videos already exist in the database.

If a viewer searches for the video of his/her interest in a video database by providing the query in the form of text in the search window, he/she is provided with a lot of irrelevant videos and NDVs in addition to the video of interest. The authors in [77] and [78] conducted an experiment of video search in Google video, Yahoo video and YouTube by giving a text-based query of 24 popular videos and found that on an average there are 27% redundant videos that are duplicate or nearly duplicate to the most popular version of a video in the search results. This again indicates that there is a strong need for the search related to query video to be ranked or ordered based on the perceptual content of the video rather than the meta-data alone.

One of the possible solutions to the above problems is to generate a short and fixed perceptual hash for a video based on its perceptual contents using methods discussed in the earlier chapters and use it for indexing the NDVs. In Chapter 2, Chapter 3, and Chapter 4, experiments were conducted to validate the various desirable properties of the hash. It was concluded that these hashes were perceptually robust descriptors of the video. Thus, to demonstrate that the hash can be used to identify perceptually NDVs existing in the database, an *near-duplicate video retrieval* (NDVR) application is developed using the video hashing algorithm based on the ARM, the TWT-ARM and finally the Tucker decomposition. To the best of our knowledge, this is the first time an NDVR application has been developed using the perceptual video hashing technique. The NDVR system, in turn, can be used for the following applications: copyright protection, video monitoring, video recommendation and video re-ranking. In addition to NDVR, the near-duplicate video detection (NDVD) [79], [80], [81], [82], [83], [84], is also an important task. NDVD is more challenging and requires significant attention, as it is the need of the hour for database cleansing, detection of copyright violation and monitoring advertisements. This work focuses on only the NDVR system and not the NDVD system.

In the literature, the perceptual video hashing also has been used for content authentication, copy detection and video tracking. In [28] and [29], the authors have used video hash based on temporally informative representative images (TIRI)s in a video copy detection system. In [45], the authors have developed a video authentication system using the temporal wavelet transform (TWT) based video hashing technique. In [85], [86] and [87], the authors have developed visual tracking application based on perceptual hashing. In [85], three following perceptual hashes, namely the average hash, the perceptive hash and the difference hash are used for video tracking. In [86] and [87], the improved Laplace-based hashing and the Laplace-based difference hashing are applied for visual tracking. The authors in [88] have integrated different features using an effective fusion-based hashing method to generate the perceptual hash vector for visual tracking applications.

The flow of the chapter is as per the following. The framework for an NDVR system is described in Section 5.1. A literature survey on video fingerprint based NDVR system is presented in Section 5.2. The scope of perceptual video hash as a video fingerprint is discussed, and then the NDVR system based on perceptual video hashing algorithms is proposed in Section 5.3. The performance of the proposed NDVR system is evaluated in Section 5.4.

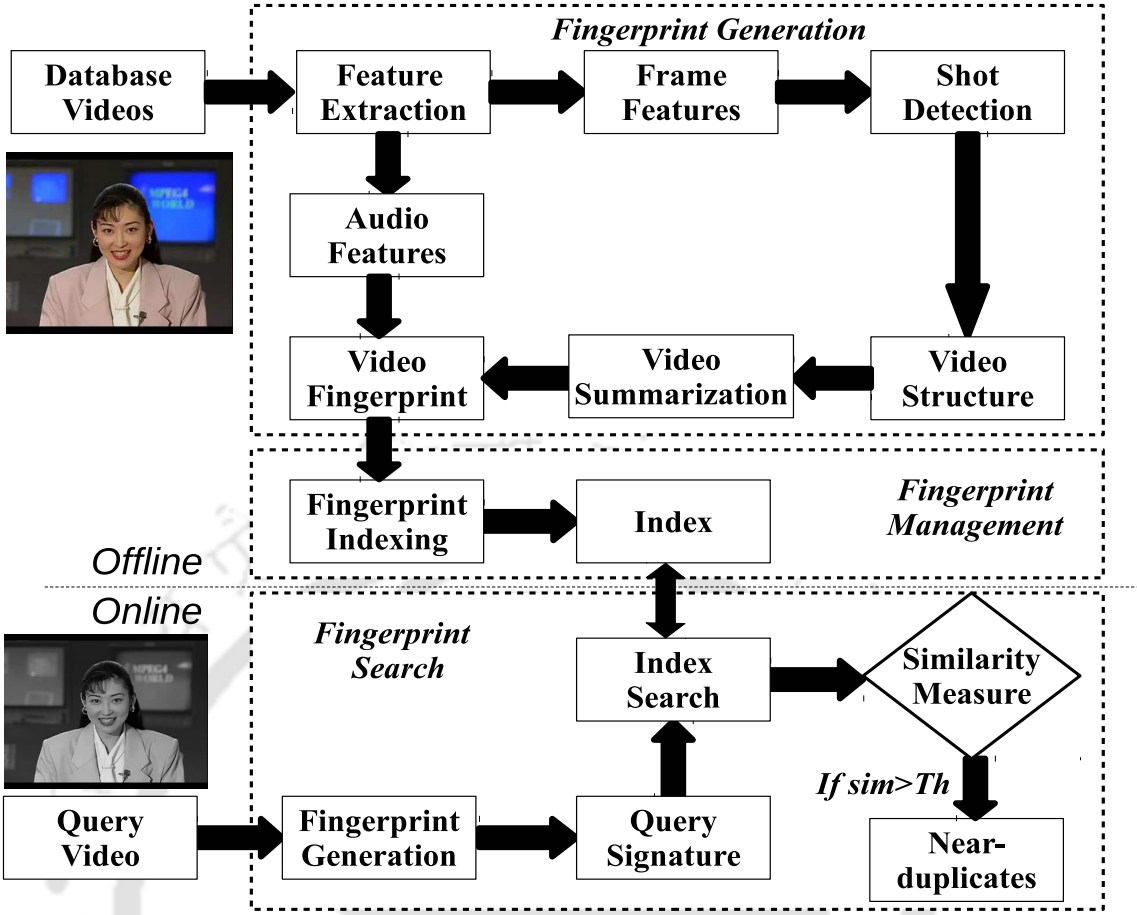


Figure 5.1: A general framework of an NDVR system.

5.1 General Framework of an NDVR System

The authors in [77] have defined the term NDV as "Identical or approximately identical videos close to the exact duplicate of each other, but different in file formats, encoding parameters, photometric variations (colour, lighting changes), editing operations (caption, logo and border insertion), different lengths, and certain modifications (frames add/remove)". Shen et al. in [89] defined the NDV as "Clips that are similar or nearly duplicate of each other, but appear differently due to various changes introduced during capturing time (camera viewpoint and setting, lighting condition, background, foreground, etc.), transformations (video format, frame rate, resize, shift, crop, gamma, contrast, brightness, saturation, blur, age, sharpen, etc.), and editing operations (frame insertion, deletion, swap and content modification)". Thus we observe that an original video undergoing content-preserving operations such as filtering, brightness modifications, contrast modifications, colour modifications, insertion of a logo, frame rate modifications and the variations in spatial resolutions are considered to be near-duplicates.

The general framework of an NDVR system is shown in Figure 5.1. The system generally consists of three major components, namely *fingerprint generation*, *fingerprint management* and *fingerprint search* as summarized by Liu et. al in [90]. These components are briefly described below.

Fingerprint generation: The fingerprint can be generated directly using the low-level features of the video. Alternatively, video summarization [91], [92] can be performed based on these low-level features and then the fingerprint can be generated from these abstract and meaningful video representations. A detailed discussion on the generation of video fingerprints is discussed in Section 5.2.

Fingerprint management: Managing the videos and their fingerprints are quite challenging as the number of videos in a database is large in number. Hence, efficient indexing structures are necessary to organize the video fingerprints and achieve faster retrieval. The fingerprint management involves identifying the efficient indexing techniques and then indexing the video fingerprints.

The objective of different methods of indexing the video fingerprint is to enable faster search and retrieval. The method of indexing depends upon the form of video fingerprint. The indexing and retrieval of multimedia data are based on the nearest-neighbour search. Different high-dimensional indexing methods exist in the literature [93], [94]. They include *tree-like structures*, *transformation methods*, and *hashing methods*. In [93], the video fingerprints are indexed using the Gauss-tree. The Gauss-tree is an index structure for managing Gaussian distributions to efficiently answer video queries [90]. The Gauss tree uses novel algorithms for query processing, insertion and tree construction. As the dimensionality increases, the feature space splitting becomes very poor with too many overlaps, and hence, the efficiency of the tree structures reduces gradually. In [94], the fingerprints are clustered and indexed based on their distance with respect to a reference point inside a cluster using a B^+- tree. The B^+- tree is an effective indexing structure for one-dimensional data. The performance of the retrieval system is dependent on the choice of the reference point. In [95], the authors have proposed an optimal choice of two far-away reference points based on the bi-distance transformation. The bi-distance transformation transforms a high-dimensional point into two distance values with respect to two optimal reference points. Another approach for indexing the video fingerprint comprises the method of locality sensitive hashing (LSH) [96], [97]. An LSH is a well-known method for indexing as it overcomes the 'curse of dimensionality' problem [96]. It hashes the fingerprints of the NDVs to the same "buckets" with high probability [98].

The fingerprint generation and indexing of database videos are usually the offline processes whereas the fingerprint search is an online process.

Fingerprint search: The query can be in the form of a *text query* or *query-by-example*. For a given query video, the fingerprint is generated and then matched with the video fingerprints in the database. If the similarity is greater than a particular threshold or if the distance between the query fingerprint and the video fingerprint in the database is less than a particular threshold, then the corresponding video is considered to be the NDV of the query video.

Some of the distance metrics include Euclidean distance, squared-chord distance, Chi-square distance, Manhattan distance and normalized Hamming distance (NHD). The NHD was discussed in Section 5.2 of Chapter 1. For discussion on the other distance measures, please refer to [99].

5.2 Survey on Video Fingerprint based NDVR System

The problem of finding the near-duplicate documents, audio and images dates back to early 1990s when the internet was evolving [100]. Finding of NDVs became a serious problem because of the size, the storage in different formats for easy downloading and streaming, evolving of video editing tools and so on. The problem of searching and retrieving NDVs was addressed in [100] through video summarization. Currently, two approaches exist in the literature for the video fingerprint generation. In the first approach, the NDVR system is developed by directly using the low-level features of the video. In the second approach, the NDVR system is developed by using the higher-level abstraction of the video called a video fingerprint, extracted from shots, keyframes or summarized video. A brief description of these approaches is presented below.

5.2.1 Low-level Features based NDVR System

Colour histogram [77], [78] and local features [101], [102] such as corners and edge pixels are used in NDVR systems. A colour histogram represents the frequency of colour pixels. An NDVR system based on it is computationally efficient and compact, but it is sensitive to variations in colour, and also it does not contain any information about the shape or texture information. Two different video frames may result in similar histograms and hence may result in the wrong retrieval. Some of the local features used in an NDVR system are blobs, corner and edge pixels. An NDVR system based on it captures the features that are usually resistant to variations in scale and different affine transformations. But the drawback is that to retrieve NDVs, a large number of features have to be compared, and hence the method becomes computationally expensive. Therefore, a small number of pertinent features, in terms of a video fingerprint, have to be extracted.

5.2.2 Video Fingerprint based NDVR System

After extracting a large number of low-level features from a video, they are operated by a fingerprint generation algorithm to reduce the dimensionality and retain the most relevant and meaningful features. The dimensionality is reduced to improve the retrieval speed. In fingerprint generation, different works focus on different aspects of the video. Depending upon the application, the fingerprints are extracted from different levels of the video, namely *frame-level local fingerprint, frame-level global fingerprint, video-level global fingerprint, spatiotemporal fingerprint* and *multi-feature fusion fingerprint*. They are outlined below.

1. **A frame-level local fingerprint**: It is extracted from the local features of the frame. Some of the local feature descriptors used are the scale-invariant feature transform (SIFT) [103], the principal component analysis (PCA)-SIFT [104] and the LBP [105]. These fingerprints are robust to complex editing, photometric variations, changes in the angle of viewport during the recording process of the same scene. These fingerprints can be used when speed is not the key requirement of an NDVR system as the method used to generate the fingerprint and implement the NDVR system are computationally expensive. Recent works have tried to increase the speed of retrieval by using the fast indexing struc-

ture techniques [82] and dimensionality reduction techniques such as *neighbour retrieval visualizer* [106].

2. **A frame-level global fingerprint**: It is extracted from the global features of the keyframes. The keyframes are extracted from the video using temporal sampling or shot boundary detection techniques. Some of the global features include the colour histogram, shape descriptors, contour representations, and texture features. Some of the recently used global feature descriptors in an NDVR system are bag-of-visual-words [107], [108] and frame-level global descriptors such as the improved Harris detector [109] and ternary frame descriptor [110]. These fingerprints are the trade-off between the speed and the accuracy of the NDVR system [90].

3. **A video-level global fingerprint**: It is extracted by treating the entire video as a single entity. The basic idea of generating this type of fingerprints is to represent the global video information into various forms, such as the histograms [77], [111], and cluster representatives [112]. These fingerprints are robust to minor colour, contrast and frame position modifications. They can be used when speed is the key requirement of an NDVR system as the methods used to generate the fingerprint and implement the NDVR system are computationally efficient [90].

4. **A spatiotemporal fingerprint**: It is extracted from the spatial and temporal features of the video. In [113], the spatiotemporal fingerprint is generated based on the conditional entropy and the local binary pattern (LBP) feature descriptors. The conditional entropy is used to select relevant features that carry as much information as possible. The LBP feature descriptor is robust against illumination changes and computationally simple, which makes it possible to extract features in challenging real-time settings. In [82], a spatiotemporal fingerprint is extracted using the pattern-based approach under the hierarchical filter-and-refine framework. In other words, the non-near-duplicate videos are filtered and removed thereby reducing the search space of the NDVs and then finally re-rank it. This method fails to retrieve the videos obtained by applying the picture-in-picture transformation [114], high-speed fast-forwarding, or super-slow motion. These fingerprints are robust to heavy colour modifications and geometric modifications but sensitive to temporal modifications on the video frame. The spatiotemporal fingerprints involve high complexity during measurement of similarity as the temporal order is taken into consideration [90].

5. **A multi-feature fusion fingerprint**: It is extracted by combining both the local and the global features of the video. The feature-level fusion is preferred over the only local feature - based or only global feature - based NDVR system because the information present in local features and global features are complementary in nature. In general, the technique is also known as multi-modal [115], [116], [117] and multiview learning [118], [119], [120] as it involves machine learning tasks and general data analysis to fuse multiple features. In [121], the video fingerprint is generated using the machine learning technique for fusing the hue-saturation-value (HSV) and LBP features extracted from the video keyframes. In

[122], [123] the fingerprint is generated based on the machine learning techniques using the convolutional neural networks. The machine learning techniques for the generation of fingerprint eliminates the need for feature extraction and selection.

Thus, there are different approaches to the generation of the video fingerprint. Currently, the fingerprints generated using the machine learning techniques are outnumbering the traditional ways of generating the video fingerprints. Whatever may be the method of generation, a video fingerprint is desired to be compact in nature to facilitate faster retrieval. Furthermore, the fingerprint generation technique should take care that all the NDVs have similar fingerprints and non-NDVs should have different fingerprints.

5.3 Proposed Framework of NDVR System

The perceptual hash of the video may be used as a fingerprint for content-based retrieving, efficient database management (by removing NDVs from the database if the perceptual contents of the videos are same), video copyright protection and video authentication. The perceptual video hash would qualify to be video fingerprint for such applications if it satisfies the following properties [2, 12, 15]:

1. It should be compact and non-invertible.
2. It should be similar for NDVs.
3. It should be dissimilar for different videos.

The properties mentioned above were validated and discussed in the earlier chapters of this thesis. Thus, the perceptual hash of the video as a fingerprint could be used to develop an NDVR system. In [124], we first demonstrated that the NDVR system could be developed using the perceptual video hash.

Figure 5.2 shows the block diagram of the application developed for the indexing and retrieval of NDVs, based on the perceptual hashing. This application retrieves the NDVs rather than relevant videos. This scheme consists broadly of two phases, namely the database creation phase and the video retrieval phase. The brief description of each phase is given below:

1. **Database creation phase**: This phase consists of two parts, namely fingerprint generation and fingerprint management. Fingerprint generation includes the preprocessing and normalization of the video, perceptual hash generation from the video and then storing the perceptual hash along with the meta-data of the video in the form of a structured array.

   The input colour videos are converted to grey videos and then normalized to a predefined size say, $\mathcal{V} \in \mathbb{R}^{X \times X \times X}$ via temporal sub-sampling and spatial resizing. This preprocessing and normalization process ensures the impact of various spatial dimensions and frame rates on the hash values to be minimal. The perceptual hash is then extracted using a perceptual hashing technique. Let, $\mathbf{h}_j; j = 1, 2, \ldots, N$ be the hash vectors for N videos $V_1, V_2, \ldots, V_N$, respectively, generated using the perceptual video hashing algorithm. Later, the videos

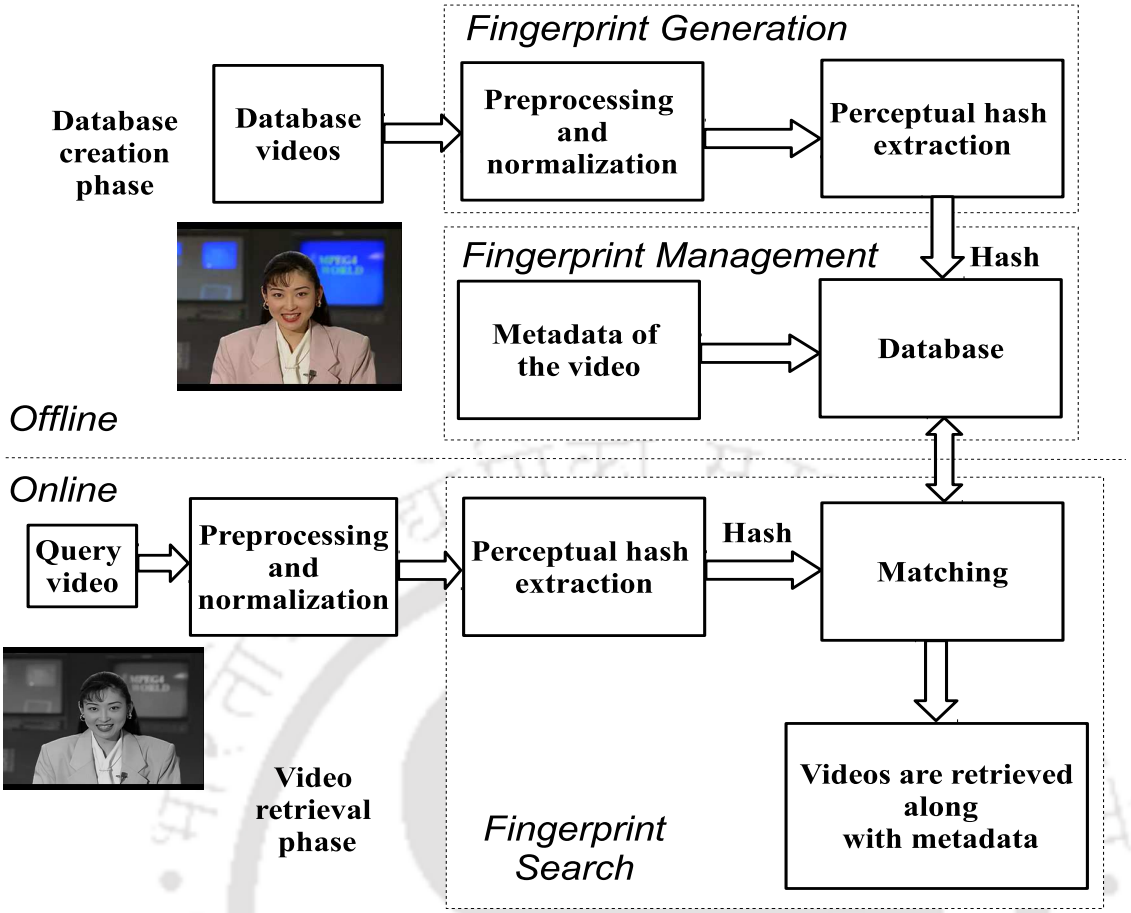


Figure 5.2: Block diagram of the perceptual hashing based NDVR application.

and the corresponding hashes $\mathbf{h}_j$ along with the meta-data of the video are stored in the database. The possible meta-data include title, owner and description. We have used the title of the video as the sole meta-data.

2. **Video retrieval phase**: In the retrieval phase, the query video is preprocessed and normalized as described in the database creation phase. The hash $\mathbf{h}_{query}$ is then computed for the query video V_{query} using the same perceptual video hashing algorithm used during the database creation phase. The $\mathbf{h}_{query}$ is then compared with all the hashes in the database using the NHD metric. The videos are considered to be NDVs and retrieved if the following condition is satisfied:

$$d_{NH}\left(\mathbf{h}_j, \mathbf{h}_{query}\right) \leq T_h\,; \quad j = 1, 2, \ldots, N \tag{5.1}$$

where d_{NH} is NHD metric and T_h is a threshold. Here, the threshold is chosen based on the experiment conducted to validate the perceptual robustness property of the hash. The detailed procedure for choosing a specific value of the threshold is described in the following section.

5.4 Simulation Results and Discussion

The following perceptual video hashing techniques are used for developing the NDVR system:

1. The Achlioptas's random projection (Chapter 3)
2. The combination of the TWT and the Achlioptas's random projection (Chapter 3)
3. The Tucker decomposition (Chapter 4)
4. The PARAFAC decomposition [15]

A typical NDVR system considers the following aspects: huge video database, scalability and efficient indexing structure to facilitate faster retrieval of NDVs. Since the prime motive of this thesis is to propose a robust perceptual video hashing algorithm and not on developing an NDVR system; we have not considered the algorithms specifically tailored for such an NDVR system for comparison. For demonstration, the perceptual hashing based NDVR system (without considering the aforementioned factors) is also developed using the perceptual video hashing algorithm based on the PARAFAC decomposition [14, 15].

Table 5.1: **Database-I** details for NDVR system using the videos downloaded from REEFVID and XIPH video databases.

Parameter	**Description**
Database	[reefvid.org] and [xiph.org]
Number of original videos	224
Number of NDVs	1792
Total number of videos	2016
Modified frame rates	15fps and 60fps
Modified spatial resolutions	704×576 and 176×144
Modified compression ratios	250:1 and 240:1
Insertion of a logo as a watermark	64×64 and 128×128

5.4.1 Database Details

A number of experiments are carried out to study the performance of the proposed NDVR system. Three different databases of different sizes are created from different sources with different levels of content-preserving and malicious attacks. The details of the databases are tabulated in Table 5.1, Table 5.2 and Table 5.3.

Table 5.1 shows the details of the **Database-I** created from REEFVID [33] and XIPH [34] video databases. In this database, 224 downloaded videos were subjected to various content-preserving operations and malicious attack (as described in Table 5.1) to create 1792 NDVs. Thus, the total number of videos including the original and near-duplicates in this set up is equal to 2016. In other words, for each original video in the database, eight perceptually close

Table 5.2: **Database-II** details for NDVR system using the videos downloaded from OPEN-VIDEO database.

Parameter	**Description**
Database	[open-video.org]
Number of original videos	224
Number of NDVs	2240
Total number of videos	2464
Modified frame rates	15fps and 60fps
Modified spatial resolutions	576×704 and 176×144
Increase in contrast	20%
Average Blurring	5×5
Adding AWGN	$\mu = 0$ and $\sigma = 57$
Cropping	10% from the boundary pixels
Insertion of a logo as a watermark	64×64 and 128×128

Table 5.3: **Database-III** details for NDVR system using the videos downloaded from REEFVID, XIPH, OPEN-VIDEO and TRECVID video databases.

Parameter	**Description**
Database	[open-video.org], [trecvid.nist.gov], [reefvid.org] and [xiph.org]
Number of original videos	1000
Number of NDVs	19000
Total number of videos	20000
Modified frame rates	15fps and 60fps
Modified spatial resolutions	576×704 and 176×144
Contrast modifications	$+100\%$ and -50%
Brightness modifications	$+5\%$ and -5%
Average blurring	5×5
Gaussian blurring	5×5
Adding AWGN	$\mu = 0$ and $\sigma = 57$
Adding salt and pepper noise	Density of 1%
Cropping	10% from the boundary pixels
Frame dropping	Only 128 frames were retained
Insertion of a logo as a watermark	64×64 and 128×128
Histogram equalization	Applied grey level histogram equalization to R, G and B frame.
Frame rotation	5° in anti-clockwise direction
Compression	Average CR of 250:1

NDVs were created. The hash was generated for each of the videos, and the corresponding meta-data (the title of the video) was stored in the form of a structured array.

Table 5.2 shows the details of the **Database-II** created from OPEN-VIDEO dataset [35]. Here, 224 downloaded videos were subjected to ten different content-preserving operations and malicious attack (as described in Table 5.2) to create 2240 NDVs. Thus, the total number of videos including the original and near-duplicates in this set up is equal to 2464.

Table 5.3 shows the details of the **Database-III** created from REEFVID [33], XIPH [34], OPEN-VIDEO [35] and TRECVID[36] video databases. Here, 1,000 downloaded videos were subjected to nineteen different content-preserving operations and malicious attack (as described in Table 5.3) to create 19,000 NDVs. Thus, the total number of videos including the original and near-duplicates in this set up is equal to 20,000.

5.4.2 Performance Evaluation

The query to the video retrieval system is presented in the form of a query-by-example. The hash is computed for the query video and then matched with the hashes in the database. The videos are then retrieved and indexed according to the match between the hash for the query video and the hashes in the database. The performance of the video retrieval system developed using all the algorithms mentioned above was evaluated using the average precision-recall curves [125, 126, 127]. The effectiveness of the retrieval systems is related to the relevance of retrieved videos in terms of precision and recall. The *Recall*, denoted by R is defined as

$$R = \frac{N_{R_{ndv}}}{N_{T_{ndv}}} \tag{5.2}$$

where $N_{R_{ndv}}$ is the *number of NDVs retrieved* and $N_{T_{ndv}}$ is the *total number of NDVs present in the database*.

It is the fraction of the NDVs that are successfully retrieved. In other words, the value of R indicates the ability of the search to find all of the NDVs in the database. If the R value of the retrieval system is close to 1, then the system can retrieve almost all of the NDVs from the database.

The *Precision*, P is defined as

$$P = \frac{N_{R_{ndv}}}{N_{R_T}} \tag{5.3}$$

where N_{R_T} is the *total number of videos retrieved*.

It is the fraction of retrieved videos that are near-duplicates to the query video. In other words, the value of P indicates the ability to retrieve only the NDVs. If the P value of the retrieval system is close to 1, then the system can retrieve only the NDVs from the database.

For each video query, the recall and the precision values are obtained. In our experiment, 50 queries were selected to obtain the average precision-recall values. These values are used to plot the average precision-recall curves. A threshold of 0.4 was used to limit the total number of NDVs retrieved out of the total number of videos retrieved. This value of threshold is chosen based on the following argument. For the video hashing algorithms under consideration, it can be observed from the validation of the perceptual robustness property (from Chapter 3 and

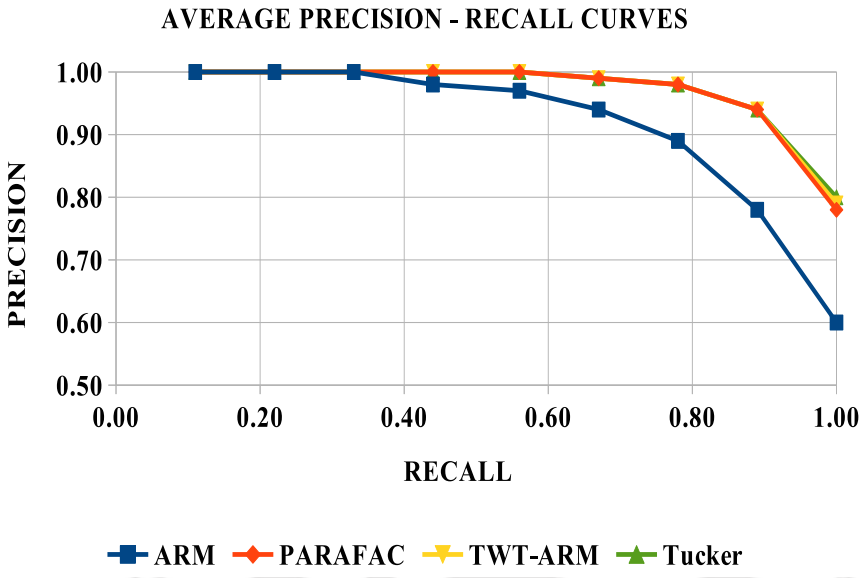


Figure 5.3: Performance evaluation in terms of average precision-recall curves for the developed video retrieval system using the proposed and the other perceptual hashing algorithms for the videos downloaded from REEFVID and XIPH database.

Chapter 4) of the hash that d_{NH} between the original video and the NDVs was found to be less than 0.5 in most of the cases. From the validation of the visual fragility property of the hash, it can be observed that the d_{NH} between the different videos ranges between 0.2 and 0.8. Therefore, to increase the retrieval of NDVs, a value of 0.4 is selected as the threshold value. The average precision-recall curves for the video retrieval system using the aforementioned perceptual video hashing algorithms are shown in Figure 5.3. The curve closest to the upper right-hand corner of the graph indicates the best performance [125, 126, 127]. From Figure 5.3, it can be observed that the performance of all the video hashing algorithms is similarly good except the video hashing algorithm based on the ARM. This is quite expected as the size of the database considered is relatively small and also from the hash validation properties and the ROC curves obtained in the earlier chapters.

Table 5.4 shows the list of first few videos retrieved for the query video titled "waterfall_cif" using the ARM based video hashing algorithm. Figure 5.5 first frame of few retrieved videos as the representative frame.

Following the same procedure described earlier, the performance of the NDVR system was evaluated on the database-II using the average precision-recall curves for 50 queries. From Figure 5.4, it can be observed that the average precision-recall curves have fallen by a small amount compared to the curves in Figure 5.3 implying that the performance of the NDVR systems has reduced by a small amount. This is due to the increase in the number of NDVs in database-II compared to the number of NDVs in the database-I. The Tucker decomposition based video hashing algorithm has shown the best performance very closely followed by the two video hashing algorithms, namely the TWT-ARM and the PARAFAC decomposition based video hashing algorithm. The performance of the ARM-based video hashing algorithm

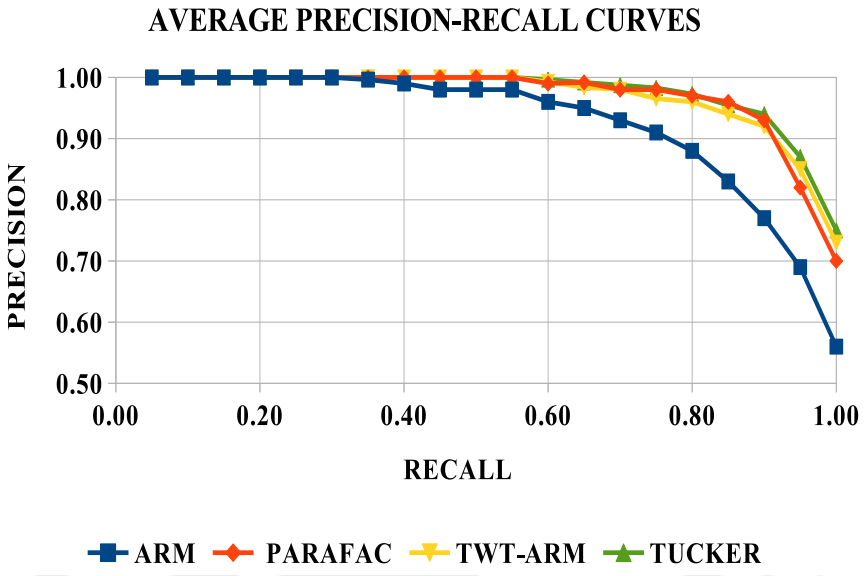
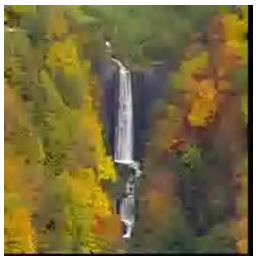


Figure 5.4: Performance evaluation in terms of average precision-recall curves for the developed video retrieval system using the proposed and the other perceptual hashing algorithms for the videos downloaded from OPEN-VIDEO database.

Table 5.4: List of first few videos retrieved for the query video titled waterfall_cif using the ARM based video hashing algorithm.

Video Retrieved	NHD
waterfall_cif	0
waterfall_cif_br100k	0.046875
waterfall_cif_br64k	0.046875
waterfall_cif_wm64	0.0625
waterfall_cif_big	0.125
waterfall_cif_small	0.125
waterfall_cif_15fps	0.140625
waterfall_cif_60fps	0.140625
clip216_wm64	0.1875

(a) waterfall_cif.



(b) waterfall_cif_br64k.

(c) waterfall_cif_wm64.

(d) clip216_wm64.

Figure 5.5: First frame of few retrieved videos as the representative frame

is relatively poor compared to the other video hashing algorithms. The argument for this performance is as follows: the ROC curves in the earlier chapters demonstrated that the Tucker decomposition based video hashing algorithm had a minimum area under the ROC curves for most of the attacks followed by the TWT-ARM, the PARAFAC and the ARM-based video hashing algorithm. Furthermore, from the hash validation properties, the overlap between the curves of perceptual robustness property and the visual fragility property was minimum for the Tucker decomposition based video hashing algorithm, indicating the maximum retrieval of NDVs; maximum for the ARM-based video hashing algorithm, indicating the minimum retrieval of NDVs.

Using the procedure mentioned above the performance of the NDVR system was evaluated on the database-III using the average precision-recall curves for 50 queries. From Figure 5.6, it can be observed that the average precision-recall curves have fallen compared to the curves in Figure 5.4 implying that the performance of the NDVR systems has reduced. This is due to considerable increase in the number of the original videos and the number of NDVs in database-III compared to the size of the database-II. It can be observed that the video hashing algorithm based on the Tucker decomposition is close to the upper right-hand corner followed by the TWT and the ARM-based video hashing, the PARAFAC decomposition based video hashing, and finally, the ARM-based video hashing. Again the performance of the ARM-based video hashing algorithm is relatively poor compared to the other video hashing algorithms. The performance of the algorithms is justified similarly on the above arguments presented for the performance of the NDVR systems on database-II.

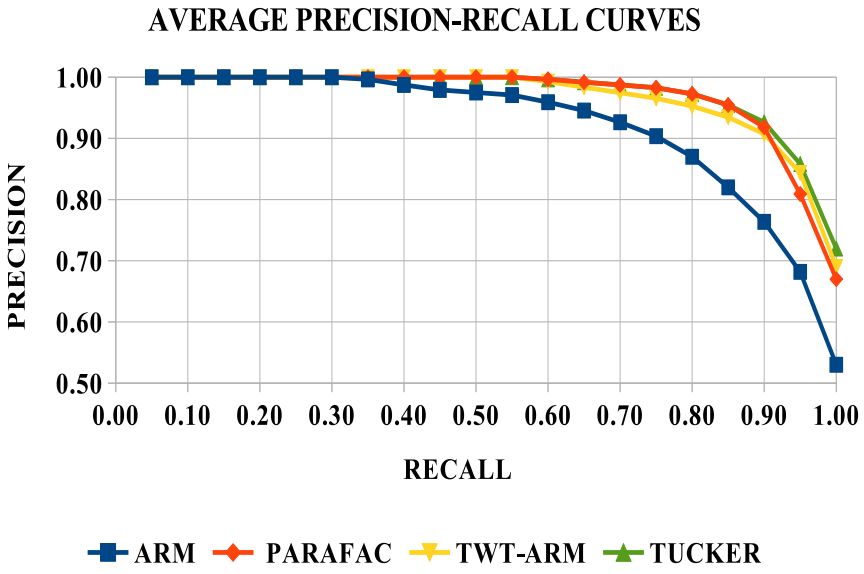


Figure 5.6: Performance evaluation in terms of average precision-recall curves for the developed video retrieval system using the proposed and the other perceptual hashing algorithms for the videos downloaded from REEFVID, XIPH, OPEN-VIDEO and TRECVID database.

5.5 Concluding Remarks

From Figure 5.3, Figure 5.4 and Figure 5.6, it can be concluded that the NDVR system based on the Tucker decomposition based perceptual video hashing algorithm is relatively stable compared to the other perceptual video hashing algorithms.

A detail discussion on the computational complexity was presented in Chapter 4. The computational complexity of all the perceptual video hashing algorithms considered for the development of the NDVR system based on the perceptual video hashing is tabulated in Table 5.5. From Table 5.5, it can be observed that the video hashing based on the ARM has the least computational complexity followed by the video hashing based on the Tucker decomposition.

Considering the retrieval performance and the computational complexity of the Tucker decomposition based perceptual video hashing algorithm, it can be concluded that this algorithm is suited best for the retrieval of NDVs among the perceptual video hashing algorithms considered.

5.6 Summary of the Chapter

This chapter is summarized as follows.

- A framework of an NDVR system and an overview of existing video fingerprint based NDVR systems were discussed.
- An NDVR system was proposed using the perceptual video hashing algorithms based on the PARAFAC, the ARM, the Tucker and combination of the TWT and the ARM.

Table 5.5: Computational complexity of the perceptual video hashing algorithms considered for the development of the NDVR system.

Name of the perceptual video hashing algorithm (PVHA)	**Computational complexity**
PARAFAC decomposition based PVHA	$\mathcal{O}\left(\frac{64}{27}X^3\right)$
Achlioptas's random projection based PVHA	$\mathcal{O}(\zeta)$
The TWT and the Achlioptas's random projection based PVHA	$\mathcal{O}(NoF \times X \times X) + \mathcal{O}\left(\frac{NoF}{2} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{4} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{8} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{16} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{32} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{64} \times X \times X\right) + \mathcal{O}\left(\frac{NoF}{128} \times X \times X\right) + \mathcal{O}(\zeta)$
The Tucker decomposition based PVHA	$\mathcal{O}(7X)$

- Three different databases were created from REEFVID [33], XIPH [34], OPEN-VIDEO [35] and TRECVID[36] databases using various content-preserving and malicious attacks.
- Average precision-recall curves were used to evaluate the performance of the proposed NDVR system.
- This curves demonstrated that the perceptual hashing algorithm can be used to develop NDVR systems.
- The performance of the NDVR system using the video hashing algorithm based on the Tucker, the PARAFAC and the TWT-ARM was found to be good. Finally, it was concluded that the perceptual video hashing algorithm based on the Tucker decomposition is suited best for the retrieval of NDVs in terms of both the retrieval performance and the computational complexity.

Chapter 6

Conclusions

Contents of the Chapter

Perceptual video hashing is an important tool for the authentication of video contents. The thesis proposed new techniques for perceptual video hashing satisfying the important properties, namely the perceptual robustness, the visual fragility and the unpredictability properties. It also explored the potential application of perceptual video hashing for the identification of near-duplicate videos (NDVs) in a database. The thesis findings are summarized below.

6.1 Summary of the Thesis

The problems addressed in the thesis can be divided into two areas:

1. Development of new perceptual video hashing algorithms for authenticating the video that has undergone common content-preserving attacks such as the addition of noise, compression and filtering.

2. Exploring the potential of perceptual video hashing as a tool for identifying the near-duplicate videos in a database.

Chapter 2, Chapter 3, and Chapter 4 addressed the first problem and Chapter 5 addressed the second. These contributory chapters can be summarized as follows:

1. **Perceptual Video Hashing based on Three Dimensional Radial Projection**
   In the 3D radial projection (3D-RASH) based video hashing algorithm, the features were

extracted from the variance of the luminance values projected along lines in a video sub-cube. This line was rotated by 4π steradians in discrete steps of 1° to cover the entire sub-cube. A sufficient number of sub-cubes were used to cover the complete video and thereby, the local and global features were extracted from the video in terms of the 3D-RASH vectors. The hash was generated using these feature vectors using two schemes. In the first scheme, these feature vectors were averaged across the rows and then projected on to the 1D-DCT basis to generate a compact hash vector. In the second scheme, the feature vectors were projected on to the 2D-DCT basis and then averaged to produce a compact hash vector. The proposed hashes based on 3D-RASH were experimentally validated for their desirable properties. The performance of the proposed algorithms was evaluated using the ROC curves and compared with the video hashing algorithms based on the PARAFAC decomposition, the 3D DCT, the CGO, and the 2D RASH. From the ROC curves, it was concluded that the performances of the proposed algorithms were satisfactory against most of the single content-preserving and malicious attacks. However, they were vulnerable to the addition of the noise.

2. **Perceptual Video Hashing based on Random Projection**
   The focus of this work was at developing a computationally efficient and simple-to-implement perceptual video hashing algorithm, and this was achieved by using the distance preserving Achlioptas's random matrix (ARM). This projection matrix consists of binary random numbers +1 and -1 and hence involves only additions and subtractions contrary to the Gaussian or the FJLT random projection matrices. Because of the computational advantage, the use of ARM-based hash in the video retrieval application results in the faster hash generation and hence faster video retrieval. In this chapter two methods were proposed to produce the perceptual hash vector from the input video. In one of the methods, the vectorized grey pixels were projected on to the Achlioptas's random basis to generate the compact hash vector. This hash vector was experimentally validated for its desirable properties. It satisfied all the desired properties of the perceptual hash. This scheme is computationally efficient, and the performance was comparable to the existing video hashing algorithms. To improve the performance further, the second method is proposed. In this method, the TIRIs were generated using the TWT. These TIRIs were vectorized and projected on to the Achlioptas's random basis to generate the compact hash vector. The characteristics of this hash improved considerably satisfying the desired properties. Furthermore, the evaluation of the algorithm using the ROC curves showed considerable improvement over the first method. Both the methods are computationally efficient.

3. **Perceptual Video Hashing based on Tucker Decomposition**
   The focus of this work was to treat the videos as tensors and generalise the video hash generation method employing the PARAFAC decomposition. We used the Tucker decomposition for the generation of perceptual video hash, because it is a generalized and a powerful tool for the analysis of the multi-way data arrays. It covers the PARAFAC model as a special case. Being a generalized tool, this model also allows the users to

select a different number of factors along each mode, during the multi-way data array decomposition, aiding better analysis. A method for finding the suitable number of components in the factor matrices of the Tucker decomposition was proposed. The hash was validated for the desirable properties of the perceptual hash, and it was found to be very satisfactory. The algorithm showed excellent robustness to most of the single and multiple attacks except the malicious attack.

4. **Application of Perceptual Video Hashing for Near-duplicate Video Retrieval** A perceptual video hashing scheme for the content-based retrieval of NDVs was proposed. The following algorithms were used to design the application: the ARM-based video hash, the TWT-ARM based video hash, the Tucker decomposition based video hash and the PARAFAC decomposition based video hash. The performance of the application was evaluated using the average precision-recall curves on three video databases created from REEFVID, XIPH, OPEN-VIDEO and TRECVID datasets using various content-preserving operations and malicious modifications. The near-duplicate video retrieval application based on the PARAFAC decomposition, the Tucker decomposition and the TWT-ARM was found to be suitable.

6.2 Future Research Directions

The thesis focused on developing perceptual video hashing techniques by projecting video into sub-spaces. There are several unexplored areas. Some of the research directions are as follows:

1. The existing video hashing methodologies in the current literature and the proposed methods utilizes only the luminance component of the video to generate the hash. It would be interesting to investigate how to make use of the colour component of the video to generate the hash.

2. In Chapter 2 of this thesis, the 3D-RASH based video hashing algorithm performed poorly against the addition of noise. The performance may be improved by considering the higher-order moments as features in addition to the variance.

3. In Chapter 3 of this thesis, the random projection based video hashing the projection matrix consisted of +1 and -1 resulting in only addition and subtraction operations. In future, it would be worth investigating the performance of the video hashing algorithms by using the combination of sparse projection matrices and fast random projection matrices.

4. In this thesis, the video hashing algorithms are proposed for generic videos and found to be robust against most of the single and multiple image processing attacks. It would be interesting to investigate how to utilize the apriori information about the characteristics of the video such as minimally changing background and then develop a robust video hashing algorithm for such videos.

5. The proposed video hashing algorithms showed strong perceptual robustness. Only the 3D-RASH based video hashing algorithm performed well against the malicious modifications of the video. All the other proposed methods perform poorly against the malicious

attacks. Hence, a direction for future research would be to design video hashing algorithms that are sensitive to malicious modifications.

6. In this thesis, the performance of the video hashing algorithms were tested on a database of 1000 videos with duration being less than 5 minutes per video due to the constraints in the availability of the computational resources. In future, the performance of the video hashing algorithms may be tested on a large video database with duration being more than 5 minutes per video.

7. The near-duplicate video retrieval application was developed in the thesis using the proposed video hashing algorithms without considering the retrieval speed. A direction for the future research would be to design a fast video retrieval application based on the perceptual video hash and test its performance on a large video database.

Bibliography

[1] B. A. E. S. Azhar Hadmi, William Puech and A. A. Ouahman, *Perceptual Image Hashing, Watermarking - Volume 2*, D. M. D. Gupta, Ed. InTech, May 2012.

[2] R. Venkatesan, S.-M. Koon, M. Jakubowski, and P. Moulin, "Robust Image Hashing," in *Image Processing, 2000. Proceedings International Conference on*, vol. 3, Sep 2000, pp. 664 –666.

[3] V. Monga, "Perceptually Based Methods for Robust Image Hashing," PhD Thesis, The University of Texas at Austin, Aug 2005.

[4] V. Monga and B. Evans, "Perceptual Image Hashing Via Feature Points: Performance Evaluation and Tradeoffs," *Image Processing, IEEE Transactions on*, vol. 15, no. 11, pp. 3452 –3465, Nov 2006.

[5] V. Monga and M. Mhcak, "Robust and Secure Image Hashing via Non-Negative Matrix Factorizations," *Information Forensics and Security, IEEE Transactions on*, vol. 2, no. 3, pp. 376 –390, Sep 2007.

[6] X. Lv and Z. Wang, "Perceptual Image Hashing Based on Shape Contexts and Local Feature Points," *Information Forensics and Security, IEEE Transactions on*, vol. PP, no. 99, p. 1, Mar 2012.

[7] S. Lee and C. Yoo, "Robust Video Fingerprinting for Content-Based Video Identification," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 7, pp. 983 –988, Jul 2008.

[8] T. Kalker, J. Haitsma, and J. C. Oostveen, "Issues with Digital Watermarking and Perceptual Hashing," in *Multimedia Systems and Applications IV*, vol. 4518. International Society for Optics and Photonics, Nov 2001, pp. 189–197.

[9] M. Bober and P. Brasnett, "MPEG-7 Visual Signature Tools," in *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*. IEEE, Jun 2009, pp. 1540–1543.

[10] C. De Roover, C. De Vleeschouwer, F. Lefebvre, and B. Macq, "Robust Video Hashing Based on Radial Projections of Key Frames," *Signal Processing, IEEE Transactions on*, vol. 53, no. 10, pp. 4020 – 4037, Oct 2005.

[11] C. D. Roover, C. D. Vleeschouwer, F. Lefebvre, and B. M. Macq, "Robust Image Hashing Based on Radial Variance of Pixels," in *ICIP (3)*, 2005, pp. 77–80.

[12] B. Coskun and B. Sankur, "Robust Video Hash Extraction," in *Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th*, Apr 2004, pp. 292 – 295.

[13] B. Coskun, B. Sankur, and N. Memon, "Spatio-Temporal Transform Based Video Hashing," *Multimedia, IEEE Transactions on*, vol. 8, no. 6, pp. 1190 –1208, Dec 2006.

[14] M. Li and V. Monga, "Desynchronization Resilient video Fingerprinting via Randomized, Low-Rank Tensor Approximations," in *Multimedia Signal Processing (MMSP), 2011 IEEE 13th International Workshop on*, Oct 2011, pp. 1 –6.

[15] M. Li and V. Monga, "Robust Video Hashing via Multilinear Subspace Projections," *Image Processing, IEEE Transactions on*, vol. 21, no. 10, pp. 4397 –4409, Oct 2012.

[16] X. Lv and Z. Wang, "Fast Johnson-Lindenstrauss Transform for Robust and Secure Image Hashing," in *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, Oct 2008, pp. 725 –729.

[17] X. Lv and Z. J. Wang, "An Extended Image Hashing Concept: Content-Based Fingerprinting Using FJLT," *EURASIP J. Inf. Secur.*, vol. 1, pp. 1–16, Jan 2009.

[18] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM Review*, vol. 51, pp. 455–500, Aug 2009.

[19] A. Cichocki, R. Zdunek, A. H. Phan, and S. Ichi Amari, *Nonnegative Matrix and Tensor Factorizations - Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, 2009.

[20] D. Achlioptas, "Database-friendly Random Projections." ACM Press, 2001, pp. 274–281.

[21] D. Achlioptas, "Database-friendly Random Projections: Johnson-Lindenstrauss with Binary Coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, Jun 2003.

[22] N. Ailon and B. Chazelle, "Approximate Nearest Neighbors and the Fast Johnson-Lindenstrauss Transform," in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. ACM, May 2006, pp. 557–563.

[23] N. Ailon and B. Chazelle, "The Fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors," *SIAM J. Comput.*, vol. 39, no. 1, pp. 302–322, May 2009.

[24] K. Hamon, M. Schmucker, and X. Zhou, "Histogram-Based Perceptual Hashing for Minimally Changing Video Sequences," in *Proceedings of the Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, ser. AXMEDIS '06. IEEE Computer Society, Dec 2006, pp. 236–241.

[25] L. Weng and B. Preneel, "A novel video hash algorithm," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, Oct 2010, pp. 739–742.

[26] F. Lefebvre, *Message Digests for Photographic Images and Video Contents*. UCL Presses, 2004.

[27] S. Lee and C. Yoo, "Video Fingerprinting Based on Centroids of Gradient Orientations," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 2, May 2006, p. II.

[28] M. Malekesmaeili, M. Fatourechi, and R. K. Ward, "Video copy detection using temporally informative representative images," in *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*. IEEE, Dec 2009, pp. 69–74.

[29] M. M. Esmaeili, M. Fatourechi, and R. K. Ward, "A Robust and Fast Video Copy Detection System Using Content-Based Fingerprinting," *IEEE Transactions on information forensics and security*, vol. 6, no. 1, pp. 213–226, Mar 2011.

[30] N. Saikia and P. Bora, "Robust Video Hashing Using the 3D-DWT," in *Communications (NCC), 2011 National Conference on*, 2011, pp. 1–5.

[31] N. Saikia and P. K. Bora, "Perceptual hash function for scalable video," *International Journal of Information Security*, vol. 13, no. 1, pp. 81–93, Feb 2014. [Online]. Available: https://doi.org/10.1007/s10207-013-0211-z

[32] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz Mappings into a Hilbert Space," in *Contemporary Mathematics, Proceeddings of the conference on Modern Analysis and Probability*, A. B. Richard Beals, Anatole Beck and A. Hajian, Eds., vol. 26, May 1984, pp. 189–206.

[33] "Test Video Sequences," 2012. [Online]. Available: http://media.xiph.org/video/derf/

[34] "Test Video Sequences," 2012. [Online]. Available: http://www.reefvid.org/

[35] "Test Video Sequences," 2016. [Online]. Available: http://open-video.org

[36] "Test Video Sequences," 2016. [Online]. Available: http://trecvid.nist.gov

[37] T. Fawcett, "An introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, Jun 2006.

[38] E. Bingham and H. Mannila, "Random Projection in Dimensionality Reduction: Applications to Image and Text Data," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01. New York, NY, USA: ACM, Aug 2001, pp. 245–250.

[39] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, ser. STOC '98. ACM, May 1998, pp. 604–613.

[40] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of Johnson and Lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, Jan 2003.

[41] S. Dasgupta, "Learning Mixtures of Gaussians," in *Proc 40th Annu IEEE Symp Foundations of Computer Science*, Oct 1999, pp. 634–644.

[42] J. T. Gill, III, "Computational Complexity of Probabilistic Turing Machines," in *Proceedings of the sixth annual ACM symposium on Theory of computing*, ser. STOC '74. New York, NY, USA: ACM, Dec 1974, pp. 91–95.

[43] L. Adleman, "Two Theorems on Random Polynomial Time," in *Foundations of Computer Science, 1978., 19th Annual Symposium on*, Oct 1978, pp. 75–83.

[44] M. Dietzfelbinger, *Primality Testing in Polynomial Time: From Randomized Algorithms to "PRIMES Is in P"*, ser. LNCS 3000. Springer, Aug 2004.

[45] N. Saikia and P. K. Bora, "Video authentication using temporal wavelet transform," in *Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on.* IEEE, Dec 2007, pp. 648–653.

[46] S. Panchanathan, M. K. Mandal, and T. Aboulnasr, "Video indexing in the wavelet compressed domain," in *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on.* IEEE, Oct 1998, pp. 546–550.

[47] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 2, no. 7, pp. 674–693, Jul 1989.

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

[49] L. Grasedyck, D. Kressner, and C. Tobler, "A literature survey of low-rank tensor approximation techniques," *GAMM-Mitteilungen*, vol. 36, no. 1, pp. 53–78, Aug 2013.

[50] B. Chen, Z. Li, and S. Zhang, "On optimal low rank Tucker approximation for tensors: the case for an adjustable core size," *Journal of Global Optimization*, vol. 62, no. 4, pp. 811–832, Aug 2015.

[51] R. Henrion, "N-way Principal Component Analysis Theory, Algorithms and Applications," *Chemometrics and Intelligent Laboratory Systems*, vol. 25, no. 1, pp. 1 – 23, Sep 1994.

[52] H. A. L. Kiers and I. Van Mechelen, "Three-Way Component Analysis: Principles and Illustrative Application," *Psychological Methods*, vol. 6, pp. 84–110, Mar 2001.

[53] D. Muti and S. Bourennane, "Multidimensional Filtering Based on a Tensor Approach," *Signal Process.*, vol. 85, no. 12, pp. 2338–2353, Dec 2005.

[54] L. Omberg, G. Golub, and O. Alter, "A Tensor Higher-Order Singular Value Decomposition for Integrative Analysis of DNA Microarray Data From Different Studies," in *Proceedings of the National Academy of Sciences*, vol. 104, no. 47, Nov 2007, pp. 18 371–18 376. [Online]. Available: http://www.sci.utah.edu/publications/omberg07/PNAS-2007-Omberg-18371-6.pdf

[55] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear Analysis of Image Ensembles: TensorFaces," in *in Proceedings of the European Conference on Computer Vision*, May 2002, pp. 447–460.

[56] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Multilinear Principal Component Analysis of Tensor Objects for Recognition," in *Proceedings of the 18th International Conference on Pattern Recognition - Volume 02*, ser. ICPR '06. Washington, DC, USA: IEEE Computer Society, Aug 2006, pp. 776–779. [Online]. Available: http://dx.doi.org/10.1109/ICPR.2006.837

[57] E. E. Abdallah, A. B. Hamza, and P. Bhattacharya, "MPEG Video Watermarking Using Tensor Singular Value Decomposition," in *Proceedings of the 4th International Conference on Image Analysis and Recognition*, ser. ICIAR'07. Berlin, Heidelberg: Springer-Verlag, Aug 2007, pp. 772–783.

[58] B. Savas and L. Eldén, "Handwritten Digit Classification Using Higher Order Singular Value Decomposition," *Pattern Recogn.*, vol. 40, no. 3, pp. 993–1003, Mar 2007. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2006.08.004

[59] Y. Liu, Y. Liu, S. Zhong, and K. C. Chan, "Tensor distance based multilinear globality preserving embedding: A unified tensor based dimensionality reduction framework for image and video classification," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10 500–10 511, Sep 2012.

[60] S. Ergin, S. Çakir, Ö. N. Gerek, and M. B. Gülmezoğlu, "A new implementation of common matrix approach using third-order tensors for face recognition," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3246–3251, Apr 2011.

[61] M. Bessaoudi, M. Belahcene, A. Ouamane, A. Chouchane, and S. Bourennane, "A Novel Hybrid Approach for 3D Face Recognition Based on Higher Order Tensor," in *International Conference on Computer Science and its Applications*. Springer, Apr 2018, pp. 215–224.

[62] Q. Li and Q. Ruan, "A Novel 3D Facial Expression Recognition Approach Based on Tensor Distance," in *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. IEEE, May 2018, pp. 169–176.

[63] Ş. Işık, K. Özkan, M. Doğan, and Ö. N. Gerek, "A Note on Background Subtraction by Utilizing a New Tensor Approach," *International Journal of Intelligent Systems and Applications in Engineering*, pp. 87–91, Dec 2016.

[64] S. Rabanser, O. Shchur, and S. Günnemann, "Introduction to Tensor Decompositions and their Applications in Machine Learning," *arXiv preprint arXiv:1711.10781*, Nov 2017.

[65] P. Baranyi, "TP Model Transformation as a Way to LMI-Based Controller Design," *Industrial Electronics, IEEE Transactions on*, vol. 51, no. 2, pp. 387–400, Apr 2004.

[66] L. R. Tucker, "Implications of Factor Analysis of Three-Way Matrices for Measurement of Change," in *Problems in measuring change*, C. W. Harris, Ed. Madison WI: University of Wisconsin Press, 1963, pp. 122–137.

[67] L. Tucker, "Some Mathematical Notes on Three-Mode Factor Analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, Sep 1966.

[68] P. Kroonenberg and J. Leeuw, "Principal Component Analysis of Three-Mode Data by Means of Alternating least squares algorithms," *Psychometrika*, vol. 45, no. 1, pp. 69–97, Mar 1980.

[69] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A Multilinear Singular Value Decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 21, pp. 1253–1278, 2000.

[70] A. H. Phan and A. Cichocki, "Analysis of interactions among hidden components for Tucker model," in *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference.* Asia-Pacific Signal and Information Processing Association, 2009 Annual , Oct 2009, pp. 154–158.

[71] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-$(r_1, r_2, \ldots, r_n)$ approximation of higher-order tensors," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.

[72] R. Ballester-Ripoll, S. K. Suter, and R. Pajarola, "Analysis of tensor approximation for compression-domain volume visualization," *Computers & Graphics*, vol. 47, pp. 34–47, Apr 2015.

[73] T. G. Kolda and B. W. Bader, "MATLAB Tensor Toolbox," 2012. [Online]. Available: https://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html

[74] P. Comon, X. Luciani, and A. L. F. de Almeida, "Tensor Decompositions, Alternating Least Squares and Other Tales," *Journal of Chemometrics*, vol. 23, no. 7, pp. 393–405, Jul 2009.

[75] I. V. Oseledets, D. V. Savostyanov, and E. E. Tyrtyshnikov, "Tucker Dimensionality Reduction of Three-Dimensional Arrays in Linear Time," *SIAM J. Matrix Analysis Applications*, vol. 30, no. 3, pp. 939–956, Sep 2008.

[76] [Online]. Available: http://tubularinsights.com/vidcon-2015-strategic-insights-tactical-advice/

[77] X. Wu, A. G. Hauptmann, and C.-W. Ngo, "Practical Elimination of Near-duplicates from Web Video Search," in *Proceedings of the 15th ACM International Conference on Multimedia*, ser. MM '07. New York, NY, USA: ACM, Sep 2007, pp. 218–227. [Online]. Available: http://doi.acm.org/10.1145/1291233.1291280

[78] X. Wu, C. Ngo, A. G. Hauptmann, and H. Tan, "Real-Time Near-Duplicate Elimination for Web Video Search With Content and Context," *IEEE Transactions on Multimedia*, vol. 11, no. 2, pp. 196–207, Feb 2009.

[79] W.-L. Zhao and C.-W. Ngo, "Scale-rotation invariant pattern entropy for keypoint-based near-duplicate detection," *IEEE Transactions on Image Processing*, vol. 18, no. 2, pp. 412–423, Feb 2009.

[80] Q. Xie, Z. Huang, H. T. Shen, X. Zhou, and C. Pang, "Efficient and continuous near-duplicate video detection," in *Web Conference (APWEB), 2010 12th International Asia-Pacific*. IEEE, Apr 2010, pp. 260–266.

[81] Q. Xie, Z. Huang, H. T. Shen, X. Zhou, and C. Pang, "Quick identification of near-duplicate video sequences with cut signature," *World Wide Web*, vol. 15, no. 3, pp. 355–382, May 2012.

[82] C.-L. Chou, H.-T. Chen, and S.-Y. Lee, "Pattern-based near-duplicate video retrieval and localization on web-scale videos," *IEEE Transactions on Multimedia*, vol. 17, no. 3, pp. 382–395, Mar 2015.

[83] Z. Zhou, Q. J. Wu, F. Huang, and X. Sun, "Fast and accurate near-duplicate image elimination for visual sensor networks," *International Journal of Distributed Sensor Networks*, vol. 13, no. 2, p. 1550147717694172, Feb 2017.

[84] Z. Zhou, Y. Wang, Q. J. Wu, C.-N. Yang, and X. Sun, "Effective and efficient global context verification for image copy detection," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 48–63, Jan 2017.

[85] M. Fei, J. Li, and H. Liu, "Visual tracking based on improved foreground detection and perceptual hashing," *Neurocomputing*, vol. 152, pp. 413–428, Mar 2015.

[86] M. Fei, J. Li, L. Shao, Z. Ju, and G. Ouyang, "Robust Visual Tracking Based on Improved Perceptual Hashing for Robot Vision," in *International Conference on Intelligent Robotics and Applications*. Springer, Aug 2015, pp. 331–340.

[87] M. Fei, Z. Ju, X. Zhen, and J. Li, "Real-time visual tracking based on improved perceptual hashing," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 4617–4634, Feb 2017.

[88] C. Ma, C. Liu, F. Peng, and J. Liu, "Multi-feature hashing tracking," *Pattern Recognition Letters*, vol. 69, pp. 62–71, Jan 2016.

[89] H. T. Shen, X. Zhou, Z. Huang, J. Shao, and X. Zhou, "UQLIPS: a real-time near-duplicate video clip detection system," in *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, Sep 2007, pp. 1374–1377.

[90] J. Liu, Z. Huang, H. Cai, H. T. Shen, C. W. Ngo, and W. Wang, "Near-duplicate video retrieval: Current research and future trends," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, p. 44, Aug 2013.

[91] A. G. Money and H. Agius, "Video summarisation: A conceptual framework and survey of the state of the art," *Journal of Visual Communication and Image Representation*, vol. 19, no. 2, pp. 121–143, Feb 2008.

[92] S. S. Thomas, S. Gupta, and V. K. Subramanian, "Perceptual video summarizationA new framework for video summarization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1790–1802, Aug 2017.

[93] C. Bohm, M. Gruber, P. Kunath, A. Pryakhin, and M. Schubert, "Prover: Probabilistic video retrieval using the Gauss-tree," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE, Apr 2007, pp. 1521–1522.

[94] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang, "iDistance: An adaptive B+-tree based indexing method for nearest neighbor search," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 2, pp. 364–397, Jun 2005.

[95] Z. Huang, L. Wang, H. T. Shen, J. Shao, and X. Zhou, "Online near-duplicate video clip detection and retrieval: An accurate and fast system," in *IEEE International Conference on Data Engineering*. IEEE, Mar 2009, pp. 1511–1514.

[96] D. Liu and Z. Yu, "A computationally efficient algorithm for large scale near-duplicate video detection," in *International conference on multimedia modeling*. Springer, Jan 2015, pp. 481–490.

[97] K. Grauman, "Efficiently searching for similar images," *Communications of the ACM*, vol. 53, no. 6, pp. 84–94, Jun 2010.

[98] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, Jun 2004, pp. 253–262.

[99] B. Patel and B. Meshram, "Content based video retrieval systems," *arXiv preprint arXiv:1205.1641*, May 2012.

[100] S.-C. Cheung and A. Zakhor, "Efficient video similarity measurement and search," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 1. IEEE, Sep 2000, pp. 85–88.

[101] X. Wu, W.-L. Zhao, and C.-W. Ngo, "Near-duplicate keyframe retrieval with visual keywords and semantic context," in *Proceedings of the 6th ACM international conference on Image and video retrieval*. ACM, Jul 2007, pp. 162–169.

[102] X. Wu, M. Takimoto, S. Satoh, and J. Adachi, "Scene duplicate detection based on the pattern of discontinuities in feature point trajectories," in *Proceedings of the 16th ACM international conference on Multimedia*. ACM, Oct 2008, pp. 51–60.

[103] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, Nov 2004.

[104] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, Jun 2004, pp. II–II.

[105] G. Zhao and M. Pietikainen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 915–928, Jun 2007.

[106] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski, "Information retrieval perspective to nonlinear dimensionality reduction for data visualization," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 451–490, 2010.

[107] Y.-G. Jiang and C.-W. Ngo, "Visual word proximity and linguistics for semantic video indexing and near-duplicate retrieval," *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 405–414, Mar 2009.

[108] L. Wang, E. Elyan, and D. Song, "Rebuilding visual vocabulary via spatial-temporal context similarity for video retrieval," in *International Conference on Multimedia Modeling*. Springer, Jan 2014, pp. 74–85.

[109] S. Poullot, M. Crucianu, and O. Buisson, "Scalable mining of large video databases using copy detection," in *Proceedings of the 16th ACM international conference on Multimedia*. ACM, Oct 2008, pp. 61–70.

[110] K.-R. Kim, W.-D. Jang, and C.-S. Kim, "Frame-level matching of near duplicate videos based on ternary frame descriptor and iterative refinement," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, Sep 2015, pp. 31–35.

[111] L. Liu, W. Lai, X.-S. Hua, and S.-Q. Yang, "Video histogram: A novel video signature for efficient web video duplicate detection," in *International Conference on Multimedia Modeling*. Springer, Jan 2007, pp. 94–103.

[112] H. T. Shen, B. C. Ooi, and X. Zhou, "Towards effective indexing for very large video sequence database," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, Jun 2005, pp. 730–741.

[113] L. Shang, L. Yang, F. Wang, K.-P. Chan, and X.-S. Hua, "Real-time large scale near-duplicate web video retrieval," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, Oct 2010, pp. 531–540.

[114] H.-J. Park, "Signal transformation system and method for providing picture-in-picture in high definition television receivers," Jan. 31 1995, US Patent 5,386,241.

[115] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, "Data fusion through cross-modality metric learning using similarity-sensitive hashing," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, Jun 2010, pp. 3594–3601.

[116] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen, "Inter-media hashing for large-scale retrieval from heterogeneous data sources," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, Jun 2013, pp. 785–796.

[117] J. Xu, V. Jagadeesh, and B. Manjunath, "Multi-label learning with fused multimodal bi-relational graph," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 403–412, Feb 2014.

[118] X. Shen, F. Shen, Q.-S. Sun, and Y.-H. Yuan, "Multi-view latent hashing for efficient multimedia search," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, Oct 2015, pp. 831–834.

[119] V. Ranjan, N. Rasiwasia, and C. Jawahar, "Multi-label cross-modal retrieval," in *Proceedings of the IEEE International Conference on Computer Vision*, Dec 2015, pp. 4094–4102.

[120] X. Liu, L. Huang, C. Deng, J. Lu, and B. Lang, "Multi-view complementary hash tables for nearest neighbor search," in *Proceedings of the IEEE International Conference on Computer Vision*, Dec 2015, pp. 1107–1115.

[121] Y. Hao, T. Mu, R. Hong, M. Wang, N. An, and J. Y. Goulermas, "Stochastic multiview hashing for large-scale near-duplicate video retrieval," *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 1–14, Jan 2017.

[122] G. Kordopatis-Zilos, S. Papadopoulos, I. Patras, and Y. Kompatsiaris, "Near-Duplicate Video Retrieval with Deep Metric Learning," in *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on*. IEEE, Oct 2017, pp. 347–356.

[123] Y. Zhang, Y. Zhang, J. Sun, H. Li, and Y. Zhu, "Learning Near Duplicate Image Pairs using Convolutional Neural Networks," *International Journal of Performability Engineering*, vol. 14, no. 1, p. 168, Jan 2018.

[124] R. Sandeep, S. Sharma, M. Thakur, and P. K. Bora, "Perceptual Video Hashing Based on Tucker Decomposition with Application to Indexing and Retrieval of Near-Identical Videos," *Multimedia Tools and Applications*, vol. 75, no. 13, pp. 7779–7797, Jul 2016. [Online]. Available: http://dx.doi.org/10.1007/s11042-015-2695-1

[125] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, Jul 2008.

[126] A. Singhal, "Modern Information Retrieval: A Brief Overview," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 24, p. 2001, Dec 2001.

[127] B. Zhou and Y. Yao, "Evaluating Information Retrieval System Performance Based on User Preference," *Journal of Intelligent Information Systems*, vol. 34, no. 3, pp. 227–248, Jun 2010.

List of Publications

Journals

1. R. Sandeep and Prabin K. Bora, "Perceptual Video Hashing based on Temporal Wavelet Transform and Random Projections with Application to Indexing and Retrieval of Near-identical Videos", *Multimedia Tools and Applications*; Springer Publications, pp. 1-21, DOI 10.1007/s11042-019-7189-0, Jan, 2019.

2. R. Sandeep, Saksham Sharma, Mayank Thakur and Prabin K. Bora, "Perceptual Video Hashing Based on Tucker Decomposition with Application to Indexing and Retrieval of Near Identical Videos", *Multimedia Tools and Applications*; Springer Publications, pp. 1-19, DOI 10.1007/s11042-015-2695-1, Jun, 2015.

Conferences

1. R. Sandeep, S. Sharma, and Prabin K. Bora, "Perceptual video hashing using 3D-radial projection technique," *Fourth IEEE International Conference on Signal Processing, Communication and Networking* (ICSCN), pp. 1-6, Mar, 2017.

2. R. Sandeep, and Prabin K. Bora, "Perceptual Video Hashing Based on the Achlioptas's Random Projections", NCVPRIPG-13, pp 1-4; DOI:10.1109/ NCVPRIPG, Dec, 2013.