Video and Camera Tamper Detection Techniques for Securing Video Surveillance Systems

A thesis submitted to University of Hyderabad in partial fulfillment

for the degree of

Doctor of Philosophy

by

Sitara K.

Reg.No. 14MCPC13

SCHOOL OF COMPUTER AND INFORMATION SCIENCES

UNIVERSITY OF HYDERABAD

HYDERABAD -500046

Telangana

India

June, 2018

CERTIFICATE

This is to certify that the thesis entitled **"Video and Camera Tamper Detection Techniques for Securing Video Surveillance Systems"** submitted by **Sitara K.** bearing **Reg. No. 14MCPC13** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by her under our supervision and guidance at IDRBT, Hyderabad.

This thesis has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma. It is also free from any plagiarism.

Parts of this thesis have been

A. published in the following publications:

1. Digital Investigation, ISSN 1742-2876, Chapter 2.
2. Forensic Science International, ISSN 0379-0738, Chapter 3.
3. Multimedia Tools and Applications, ISSN 1573-7721, Chapter 6.

B. presented in the following conferences:

1. 2017 IEEE 13th International Colloquium on Signal Processing & its Applications (CSPA), Batu Ferringhi, Malaysia, International
2. Symposium on Signal Processing and Intelligent Recognition Systems (SIRS-2015), IIITMK, Trivandrum, International

Further, the student has passed the following courses towards fulfillment of coursework requirement for Ph.D:

	Course Code	Name	Credits	Pass/Fail
1	BT701	Data Structure and Algorithms	4	Pass
2	BT702	Operating System and Programming	4	Pass
3	BT704	Image Processing & Computer Vision	4	Pass
4	BT707	Cyber Forensics	4	Pass

Supervisor
Prof. B. M. Mehtre
IDRBT
Hyderabad-500 057
India

Director
Dr. A. S. Ramasastri
IDRBT
Hyderabad-500 057
India

Dean
Prof. Arun Agarwal
SCIS, UoH
Hyderabad -500 046
India

DECLARATION

I, **Sitara K.**, hereby declare that this thesis entitled **"Video and Camera Tamper Detection Techniques for Securing Video Surveillance Systems"** submitted by me under the guidance and supervision of **Prof. B. M. Mehtre**, is a bonafide research work and is free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Librarian is enclosed.

Date:

Signature of the Student

(Sitara K.)

Reg. No.: 14MCPC13

//Countersigned//

Signature of the Supervisor:

Dedicated To My Family & Teachers

Acknowledgements

I am deeply indebted to **Prof. B. M. Mehtre**, my thesis advisor, for his guidance, wisdom, and support during the course of my Ph.D. at IDRBT. I am appreciative of the time and effort he has devoted to advising me over the past years, and his consistent encouragement and positive reinforcement have made it a gratifying experience. I have learned from him the virtues of rigor and professionalism. I am grateful to him for having kept me focused on the times it mattered a lot. Besides my advisor, its my pleasure and privilege to have associated with **Prof. B. L. Deekshatulu**.

I would like to thank my Doctoral Review Committee members: **Prof. V. N. Sastry** and **Dr. M. V. N. K. Prasad** for their encouragement, insightful comments, and hard questions.

It is my privilege to thank **Dr. A.S. Ramasastri**, Director, IDRBT for reviewing the work and timely suggestions throughout my research. for his considerate support and encouragement throughout the tenure of my research work, for allowing me to utilize so much of valuable time for the research and letting me use the lab facilities and extensive library resources at institute campus. I also thank **Mr. B. Sambamurthy**, Former Director, IDRBT for extending his cooperation at the preliminary stage of my research. I also want to thank **Prof. Arun Agrawal**, Dean, School of Computer and Information Sciences (SCIS), University of Hyderabad, Hyderabad for his academic support throughout research work.

My sincere thanks also goes to **Prof. Venu Govindaraju**, **Dr. P. Ananth Raj**, **Prof. D. K. Subramanian** and **Prof. B.Yagna**

Narayana for critically reviewing my work periodically and suggesting improvements.

I am humbled to have interacted with **Prof. Atul Negi**, faculty in University of Hyderabad for constant support.

I thank **Dr. G. R. Gangadharan**, **Prof. V. Ravi**, **Dr. M. V. Sivakumaran**, **Dr. N. P. Dhavale**, **Dr. Rajarshi Pal**, **Dr. V. Radha**, **Dr. Raghu Kishor Dr. N. V. Narendra Kumar** and **Dr. P. Syam Kumar**, faculties at IDRBT.

I would like to express my sincere gratitude to **Shri G. Raghuraj**, **Shri Vijay Belurgikar**, **Dr. S. Rashmi Dev**, **Shri K. V. R. Murthy**, **Smt. Varsha Srivastava**, **Shri K. Srinivas**, **Smt. R. Jayalakshmi**, **Shri V. S. Mahesh**, **Shri. Srinivas P. Ratnakumar** and **Shri Patrick Kishore** at IDRBT.

I like to thank my teacher and philosopher at Vimala Hridaya Girls H. S. S. Kollam, **Rev. Mother. Ambika Mary** for her constant motivations and support. I am grateful to my teachers at College of Engineering, **Dr. Sreelakshmi R.** and **Dr. Anvar A.** who motivated me in research and higher studies. I thank my teacher at Govt. Model H. S. S. **Mr. Ramachandran G.**.

I extend my gratitude towards my faculties **Dr. Wilscy M**, **Dr. Madhu S. Nair** and **Ms. S. Remya** at Department of Computer Science, University of Kerala, Thiruvananthapuram.

During the course of my Ph.D., I was supported in countless ways by a large number of people. Therefore I would like to take this opportunity to thank my family, friends and colleagues and mention the following people explicitly:

I would like to thank my senior batch Ph.D candidates, especially, **Dr. Hiran V Nath**, who helped in exploring latest areas of security and forensics. Also, I thank **Dr. Ghanshyam Bopche** for being supportive in lab activities. I also thank **Gopal Narayan Rai**,

B. Shravan Kumar, **Chandra Sekhar**, **Pradeep Kumar**, **Ilaiah Kavati**, **Sriramulu Bojjagani**, **Deepnarayan Tiwari**, **Anup Kumar Mourya**, **Kumar Ravi**, and **M. Sandhya**. My colleagues- **Gutha Jaya Krishna**, **S. K. Kamaruddin**, **V. Dinesh**, and **Ravi Uyyala**. My colleagues of the junior batch- **P. Avinash**, **K. Suresh**, **Malvika Singh**, **Shadab Ahmad**, **P. Praveen Kumar**, **G. Jaya Rao**, **John Paul C. I**, and **Venkata Subramanyam K**.

My friends in University of Hyderabad, **Renny**, **Suresh Shanmughan**, **Chanthu**, **Vineetha**, **Jasna Jayaraj**, **Lidya**, **Manojan**, **Greeshma Gopinath** and **Greeshma Justin** who provided a pleasant hospitality in the campus.

I thank my friends **Annes Philip**, **Ameera P. M.**, **Shreeja R**, **Aneesh**, **Mohamed Irshad**, **Praveen Baby**, **Liji Ashish**, and **Ashish S** for being in touch always for anything.

I also thank our lab members of the present and past, especially, **Soujjanya**, **Kranthi**, **Vamsee Krishna**, **Suresh**, **Charles Libin** and **Ganesh**. I am grateful to all the participants of the cyber security drills that I was co-ordinating.

Finally, I acknowledge the people who mean a lot to me, my parents (**S. Komalavally** and **K. Sathyadevan**), in-laws (**A. C. Usha** and **V. K. Thankappan**), grandmother (**Sarasamma**), niece (**Ayaana R.**), sisters (**Kusum K.** and **Monisha Chandran**), brothers (**Abhijith S.** and **Aravind S.**), and brother-in-law (**Vishnuprasad V. T.** and **Riaz Ahmed**). I salute you all for the selfless love, care, pain and sacrifice you did to shape my life.

This last word of acknowledgment I have saved to a very special person, my husband, **Manu V. T.** who has been with me all these years and has made them the best years of my life. You were always around at times I thought that it is impossible to continue, you helped me to keep things in perspective.

Abstract

Video surveillance systems are vital source of information in crime investigations. A video surveillance system should record videos with correct field of view and be of good quality to support decision making. Otherwise, it may not be suitable for forensic investigation or other analysis purposes. Videos are acceptable as evidence in the court of law, provided its authenticity and integrity are scientifically validated. To conceal the shady activities, perpetrators may tamper the recorded video or the camera itself. The visual content of surveillance videos can be altered locally or remotely. Such malicious alterations of video contents (called video tampering) are categorized into inter-frame and intra-frame tampering.

In this thesis, we propose inter-frame video tampering detection techniques. Various inter-frame video tampering include frame insertion, frame deletion, frame duplication and frame shuffling. For tamper detection, we use tamper traces from spatio-temporal and compressed domains. Videos containing frames recorded as part of sudden camera zooming activity may get misclassified as tampered. We propose a method for zooming detection which when incorporated into video tampering detection method will address this issue. The proposed method is capable of differentiating various inter-frame tamper events and its localization in the temporal domain. Frame shuffling detection, hitherto an unexplored area in video tampering detection, is also addressed in our work. The proposed method is tested on a custom built dataset containing 23586 videos of which 2346 are pristine and rest of them are candidates of inter-frame tampered videos. It

is evident from experimental results that our method is effective in detecting the four types of inter-frame video tampering.

Synthetic zooming, which is not a state-of-the-art video tampering technique, can be used for video content manipulation. It is performed by upscaling individual frames of a video at various scales followed by cropping them to original frame size. These manipulated frames resemble genuine natural camera zoomed frames and hence may be misclassified as a pristine video by video forgery detection techniques. Even if such a video is classified as forged, forensic investigators may ignore the results believing it as part of natural camera zooming activity. Hence, this makes it suitable for a good anti-forensic method which aims to hide or eliminate digital evidence. We have proposed a method for differentiating natural camera zooming from synthetic zooming during video tampering detection using pixel variance correlation and sensor pattern noise. Experimental results on a custom built dataset containing 3200 videos show the effectiveness of the proposed method.

Generally, surveillance systems are unmanned due to limitations of manual monitoring. Considering the difficulty of getting access to storage media and the expertise in tampering video, perpetrators may prevent the surveillance camera from recording its intended field of view. Because of these, automated analysis of live video feed and automatic detection of suspicious activity have recently gained importance. We have proposed methods for detecting camera tampering events such as occlusion, defocus and displacement in static and panning surveillance systems. Effectiveness of our methods are tested on public datasets for camera tampering detection in static and panning surveillance systems. The results obtained are encouraging with a high detection rate of camera tampering and low false alarm rates. The proposed method can automatically detect routine problems with surveillance cameras like dirt on a camera lens, fog, and smoke.

Glossary

AC Alternating Current.

AGOP Adaptive Group Of Pictures.

B-frame Bi-directionally predicted frame.

BBVD Block-wise Brightness Variance Descriptor.

CBR Constant Bit Rate.

CCA Canonical Correlation Analysis.

CCD Charge-Coupled Device.

CCTV Closed-Circuit Television.

CFA Cross-modal Factor Analysis.

CHD Color Histogram Difference.

CMOS Complementary Metal-Oxide Semiconductor.

crf Constant rate factor.

DCT Discrete Cosine Transform.

DFT Discrete Fourier Transform.

DP Deletion Point.

DVR Digital Video Recorder.

EM Expectation-Maximization.

ESD Generalized Extreme Studentized Deviate Algorithm.

FFT Fast Fourier Transform.

FLANN Fast Library for Approximate Nearest Neighbors.

FLD Fisher's Linear Discriminant.

FN False Negative.

FNR False Negative Rate.

FOV Field Of View.

FP False Positive.

FPR False Positive Rate.

fps Frames per second.

GCAP Group Coherence Abnormality Pattern.

GMM Gaussian Mixture Model.

GOP Group Of Pictures.

HMRF Huber Markov Random Field.

HOG Histogram of Oriented Gradients.

HSV Hue-Saturation-Value.

I-frame Intra-coded frame.

I-MB Intra-coded Macro-Block.

JPEG Joint Photographic Experts Group.

k-NN k-Nearest Neighbors.

k-SVD k-Singular Value Decomposition.

LBP Local Binary Pattern.

LIBS Local Illumination based Background Subtraction.

LSA Latent Semantic Analysis.

MACE-MRH Minimum Average Correlation Energy Mellin Radial Harmonic.

MAP Maximum A Posteriori.

MB Macro-Blocks.

Mbps Megabits per second.

MCEA Motion-Compensated Edge Artifact.

MLS Multi-Level Subtraction.

MPEG Moving Picture Experts Group.

MV Motion Vectors.

NLF Noise Level Function.

OF Optical Flow.

OPVF Optical Flow Variation Factor.

P-frame Predicted frame.

P-MB Predicted Macro-Block.

PETS Performance Evaluation of Tracking and Surveillance.

PIV Particle Image Velocimetry.

PTZ Pan-Tilt-Zoom.

QCCoLBP Quotients of consecutive Correlation Coefficients of Local Binary Pattern.

QP Quantization Parameter.

qs Quantization scale.

RANSAC RANdom SAmple Consensus algorithm.

RI Relocated I-frames.

RMSE Root Mean Squared Error.

ROI Region Of Interest.

S-MB Skipped Macro-Block.

SARP Sequence of Average Residual of P-frames.

SCI Source Camera Identification.

SF Scale Factor.

SIFT Scale Invariant Feature Transform.

SPN Sensor Pattern Noise.

SROI Spatial Region Of Interest.

SSE Sum of Squared Errors.

SULFA Surrey University Library for Forensic Analysis.

SURF Speeded Up Robust Features.

SVD Singular Value Decomposition.

SVM Support Vector Machine.

TN True Negative.

TP True Positive.

TPM Transition Probability Matrix.

VAP Visual Analysis of People.

VBR Variable Bit Rate.

VFI Velocity Field Intensity.

VPA Variation of Prediction Artifact.

VPF Variation of Prediction Footprint.

ZOCM Zernike Opponent Chromaticity Moments.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Video surveillance systems are increasingly being used to secure public and private infrastructure, lives of citizens and high security places. A paradigm shift is observed from the use of static cameras to Pan-tilt-zoom (PTZ) cameras over the years due to additional features like capturing a large field of view (FOV) and, control of zoom and direction remotely. Surveillance cameras have become inevitable part of crime investigation and are even described as "God's eye" in certain cases. Closed-circuit television (CCTV) footages are vital source of information in crime investigations. It may help in understanding how and when the incidents have occurred, in addition to probable identification of who all were involved.

A recent study on the efficacy and efficiency of surveillance systems in crime investigation by Asbhy [6] tried to find out how often CCTV provides useful evidence and how this is affected by circumstances. This study analyzed 251195 crimes recorded by British Transport Police which occurred on British railway network between 2011 and 2015. Out of 45% of cases where CCTV footage was available, it was useful in 29% which is 65% of it. CCTV footages contributed substantial increase in solving most type of offenses. In this, some crimes are very unlikely to be solved when CCTV was not available. Though this research was conducted on CCTV footages from railways, it conveys that CCTV footage is a powerful tool for crime investigation and investigators should make use of it. Some investigators may request CCTV footage in more-serious cases only, but the usefulness of CCTV footage in solving cases is not associated with the degree

of seriousness. It may be possible that there are unresolved cases where CCTV footages might be useful in solving it, but investigators have not seized the CCTV and vice versa.

A block diagram of CCTV is shown in Fig. 1.1. The recordings of CCTV are stored in Digital Video Recorder (DVR). Authorized persons can access DVR through the display and control unit directly attached to it or through intranet which is confined to the infrastructure. They may also access DVR through remote connection if it is connected to internet.

Figure 1.1: Block diagram of CCTV

The purpose of video surveillance systems can be defeated, in general, in two ways:

1. Tampering during recording: Perpetrators can physically attack the surveillance cameras (camera tampering discussed in Section 1.3) so that the videos recorded during this attack may not contain the exact visuals of the intended FOV. Apart from camera malfunctioning, in extreme cases, physical damages may also lead to the complete failure of the system. These types of attacks may leave clues of tampering which may be visible in prima facie.

2. Tampering recorded videos: Altering stored videos (video tampering discussed in Section 1.1 and 1.2) may rarely catch the attention of a person in charge of a surveillance system. For successful execution of this tampering, perpetrators should have expertise in network intrusion and video editing.

The first approach may be successful only if the following conditions exist: (i) there is no operator for real-time monitoring of surveillance videos; (ii) if an operator exists, then perpetrator in successful in distracting his/her attention; and

(iii) no mechanism for real-time automatic camera tampering detection. Considering the risks involved in the first approach, the second one is somewhat simple if a perpetrator has the right tools. Technological advancements have contributed free and simple tools to intrude into network systems and perform video tampering for accomplishing the task stealthily. This approach deals with storage media of video surveillance systems within an organization based on its susceptivity towards virtual or physical access by perpetrators. Perpetrators can perform this in 2 possible ways depending on his/her privileges.

1. *An insider with sufficient information about an organization and access privileges.* He/she may be working within the premises of organization and may even directly access the device which stores the recorded video.

2. *An outsider without much information and less privileges.* He/she has to intrude into the system which is connected to the storage media by exploiting network vulnerabilities. Perpetrators have to overcome the hurdles of possible firewalls and other perimeter security devices deployed. But organizations rarely deploy such defense mechanisms.

Another access method to the storage media is via a genuine remote access service of the device with stolen credentials. Attackers in this case can be an outsider or insider. Figure 1.2 presents the above 2 approaches for getting access to DVR for video tampering. It shows that an insider can directly access the system which is connected to DVR while outsider accesses the system through the internet. These access paths are represented using red directed lines in the figure. The symbols which appear like thunderbolt represents the attack. An attacker (insider/outsider) physically attacking the cameras for disrupting the recordings during criminal activities is also indicated.

1.1 Video Tampering

Perpetrators can manipulate contents of video recorded during criminal activities easily using latest video editing software tools to mislead investigation. A

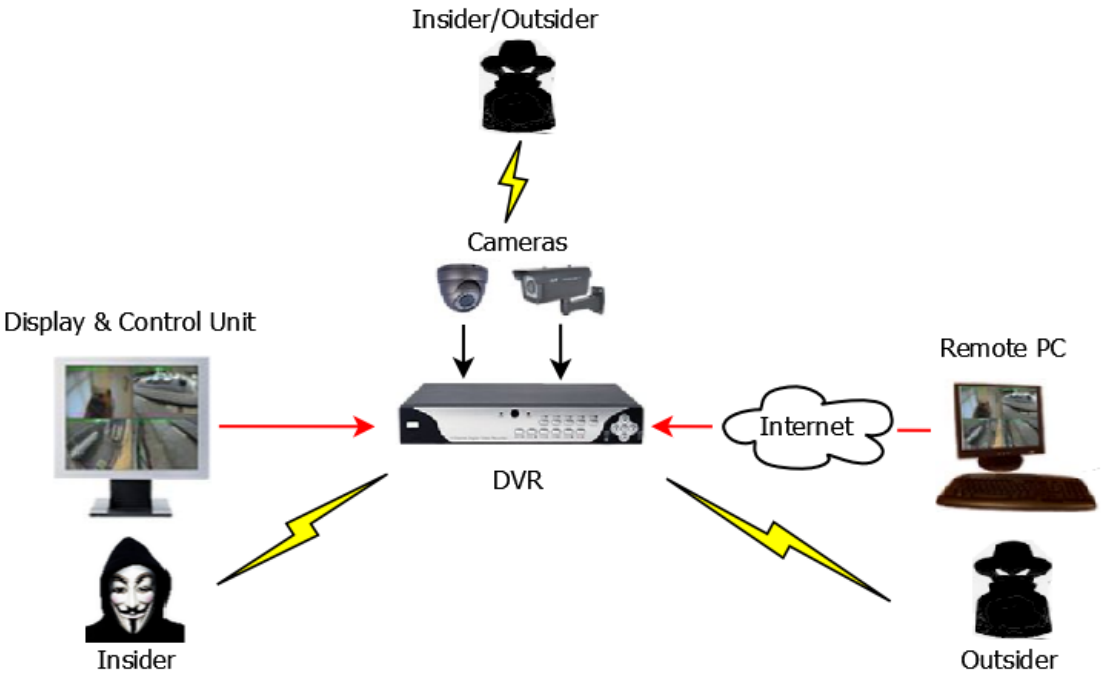


Figure 1.2: Attack paths for tampering video surveillance systems

content-manipulated video can deceive the viewer and is also capable of influencing the viewers' thoughts and impressions about recorded events - "Seeing is no longer believing". The process of changing the content of a video so as to conceal or introduce objects or events in the video is called video forgery (also known as video tampering). Verifying the authenticity of such videos is a challenge for mass media and investigating agencies. Tampered videos have the power to influence political views and provoke people in a peaceful society which may, in turn, disrupt the law and order situations. Identifying a tampered video by visual analysis is difficult or impractical. Hence, credibility of such videos becomes questionable for digital evidence. Videos are acceptable as evidence in the court of law, provided its authenticity and integrity are scientifically validated otherwise it could be challenged by defense attorney. For restoring the public trust towards video by revealing the absence or presence of tampering activity, video forensics gained significant research interest in recent times. Video forensics is the scientific analysis of a video for identifying and extracting evidences to verify its authenticity, integrity or both. In the literature of video forensics, identifying the existence

of traces of intentional or unintentional editing is the primary goal, followed by locating the areas of tampering. Video editing operations leave traces of editing called tamper artifacts or tamper footprints in a video. The authenticity of a video thus created can hence, be examined using these tamper footprints.

Perpetrators who are aware of video tampering detection methods can erase or falsify tamper traces left by editing operations to mislead forensic analysis. The techniques designed for this are called anti-forensic or counter forensic techniques. Anti-forensics itself may result in new tamper traces. Hence, forensic investigators should analyze video files considering the possibility of anti-forensic operations been applied.

Video tampering detection is a subcategory of video forensics which examines the video for content alterations and may locate the spatial or temporal locations of forgery. Video tampering detection methods can be classified as active and passive methods based on the availability of prior information. Active methods are constrained by the fact that it requires pre-embedded information in the file under investigation like a digital signature and watermarking [2, 11, 22, 23, 76, 85, 92, 119, 128] for its legitimacy testing. For a forensic analyzer, anticipation of such a file with a verifiable fingerprint is a most unlikely event. These methods may not work in conditions where video tampering is performed prior to inserting digital signature or watermark. Passive methods suit real forensic investigation scenarios as it does not require prior information and in turn utilizes the inherent properties to carryout forensic analysis. Many of the videos floating on internet in general and social media platforms in particular, are subjected to seemingly innocent content alterations, compression, meta-data removal etc as part of their standard operating procedures. This may remove passive forensic traces and has become a challenge for investigators.

Video tampering can be classified into two categories: (1) inter-frame and (2) intra-frame. In the former, an entire frame undergoes tampering. On the contrary, in the latter, manipulation takes place only on a few portions of the frames. In our work, the focus is on inter-frame video tampering which can be further classified as:

- **Frame deletion:** A sequence of frames in a video is removed to erase an event.
- **Frame insertion (or addition):** It involves copying a sequence of frames from one video and pasting them in another video.
- **Frame duplication (or replication):** It comprises copying a sequence of frames from a video and placing them at a different temporal position in the same video.
- **Frame shuffling:** It is a type of frame duplication where the sequential order of copied frames are re-arranged temporally before placing them at a different temporal position in the same video.

Frame insertion, duplication or shuffling serve as a gap filler for the deleted frames in a video. Frame shuffling may be regarded as an anti-forensic strategy for countering the detection of frame duplication. Candidate frame sequence involved in frame shuffling is reordered in such a way that the original frame order is not followed. On static surveillance systems, frame shuffling is a successful tampering technique provided the visuals of a candidate sequence of tampering contain only non-moving foreground objects or none of them. Unlike panning surveillance recordings, frame shuffling will not disturb the continuity of a video recorded by a static camera. It appears genuine to an examiner as he/she may not see any discontinuities while watching the video. In general, frame duplication detection methods look for repetitions of the frame sequence or its corresponding feature representations in a particular order. If the frames undergo reordering, then this condition never holds.

The pictorial representation of various inter-frame forgeries are illustrated in Fig. 1.3. Figure 1.3(a) shows an original video sequence consisting of 9 frames. Figure 1.3(b) shows frame deletion where frames outlined with dotted lines correspond to deleted frames. Frame deletion causes shifting of 7^{th} frame to the 4^{th} position. Figure 1.3(c) shows frame duplication where frames 1 to 3 are copied and replicated after 5^{th} frame. Frame duplication is performed after deleting frames 6 to 8 from the replicated positions. Frame shuffling is shown in Fig. 1.3(d) where frames 5 to 8 in the original sequence are replaced with frames from

1 to 4 after rearranging them. Figure 1.3(e) represents frame insertion forgery, where frames in blue colour are the newly added frames taken from a different video. As insertion is performed without deleting any frames from original video, it will cause shifting of frames in forged video after the inserted frames according to the number of frames inserted. The 4^{th} frame in original will now appear at 7^{th} frame position and so on. The actual positions of these shifted frames in original are written inside the frames.

Let us discuss a robbery scenario occurring on a FOV being recorded to understand how perpetrators can utilize inter-frame video tampering for deceiving or deviating the investigation team. Perpetrators can erase the robbery visuals by deleting the corresponding frames. Then, he/she can fill the frame gap due to frame deletion using a sequence of frames from the same video or another video. If the background of this frame sequence is same as that of the deleted frames and contains the visuals of a person 'X' entering the crime scene, then it may append the additional information of the involvement of person 'X' in a robbery. These instances can have a severe impact on the timeline analysis of events in case investigation. It may create confusions and contradictions.

1.2 Synthetic Zooming

A video can also be manipulated using synthetic zooming without using the state-of-the-art video forgeries. Synthetic zooming is performed by upscaling individual frames of a video with varying scale factors (SF) followed by cropping them to original frame size. These manipulated frames resemble genuine natural/optical camera zoomed frames and hence may be misclassified as a pristine video (a video without content manipulation) by video forgery detection algorithms. Even if such a video is classified as forged, forensic investigators may ignore the results believing those frames are recorded as part of natural camera zooming activity. Hence, this make it suitable for a good anti-forensic method which aims to hide or eliminate digital evidence.

Let us consider two examples to understand synthetic zooming forgery. A perpetrator has to delete the frames corresponding to an event. First, he/she can fill the gap of deleted frames by frame duplication, insertion or shuffling. If

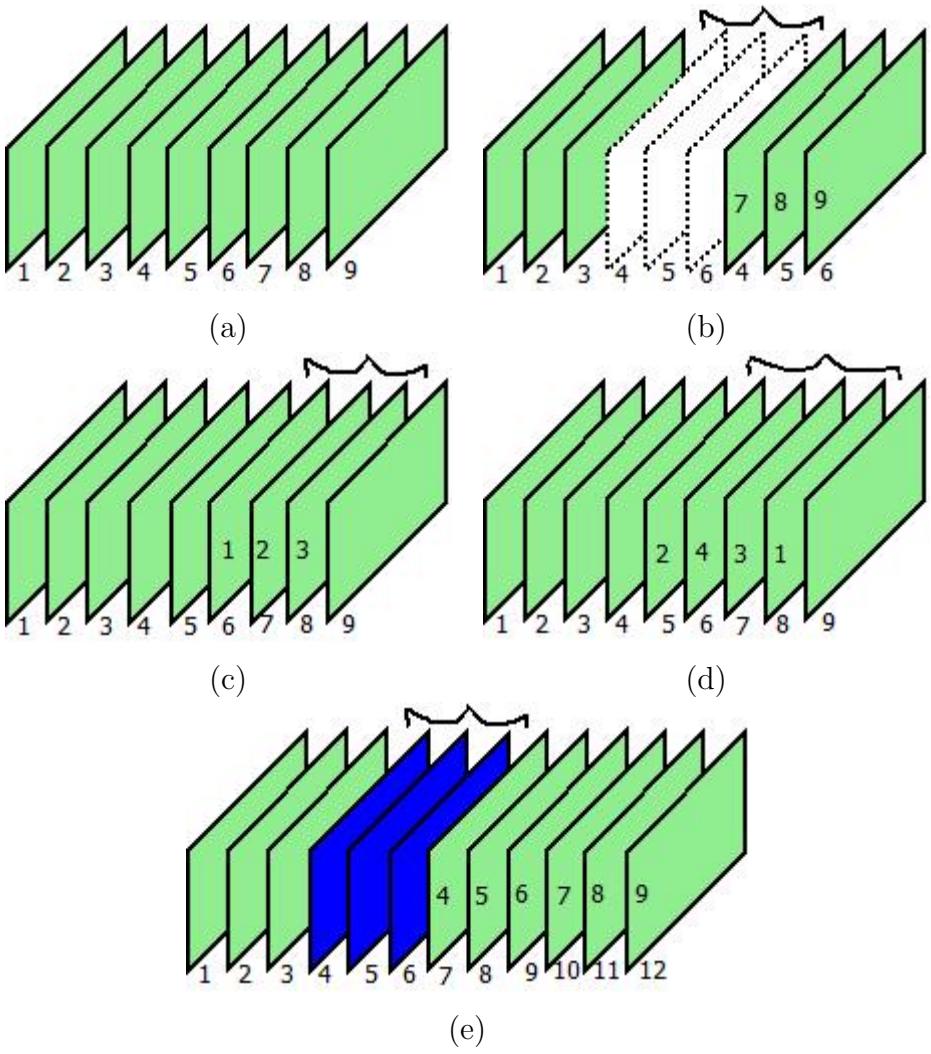
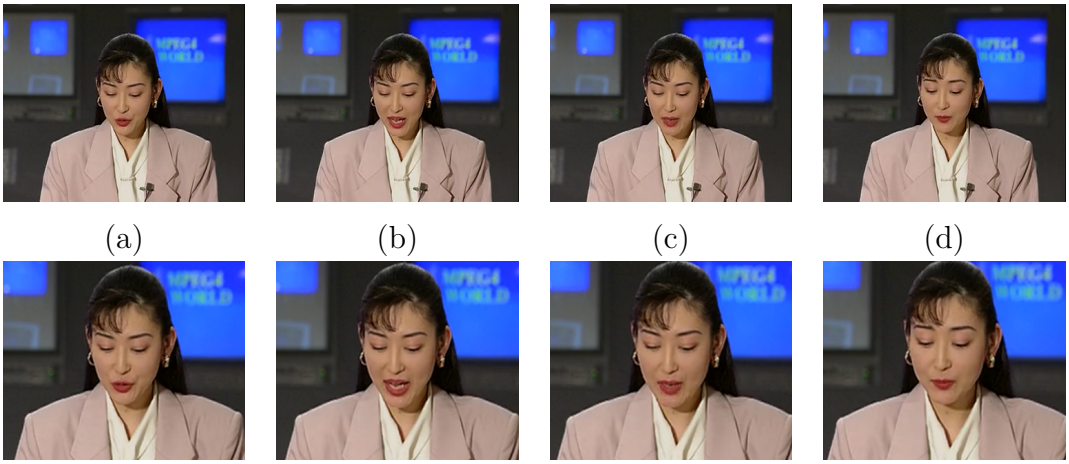


Figure 1.3: Pictorial representation of video inter-frame forgery. (a) Original Video; (b) Frame Insertion; (c) Frame Deletion; (d) Frame Duplication and (e) Frame Shuffling

these frames used to cover deletion are upscaled with a constant SF, x, and are cropped to match the original frame size of video, then it may get detected as frame insertion irrespective of the source of manipulated frames (from the same video or not). The manipulated frames may be from the same video which has to detected as frame duplication or shuffling, but methods in [50, 102, 142] fail due to upscale-crop forgery. A rectangular strip of area is lost from the boundary regions

of intended FOV. It makes the features of replicated frames different from that of its original. Second example, perpetrator can erase that event without even deleting its corresponding (candidate) frames using upscale-cropped frames. The candidate frames are upscale-cropped with a constant SF, x, such that the region corresponding to the event is not present in video after cropping. The frames in sub-sequence that immediately precedes the candidate frames are manipulated (upscale-cropped) in such a way that SF, y, $(1 < y \leq x)$ is increasing gradually until it reaches x. Similarly, a small sub-sequence that immediately follows the candidate frames is upscale-cropped gradually with decreasing SF, z, $(x \geq z > 1)$ starting from x. These synthetically created frames resemble natural camera zoom-in and zoom-out operations.

Video editing tools such as Blender, Adobe Premiere Pro, etc can be used to create tampered videos with synthetic zooming. This kind of tampering creates forged frames that merge seamlessly with the pristine frames in video without introducing any visual distortion in tamper sequence. Sample frames taken from synthetically zoomed video created using Adobe After Effects are given in Fig. 1.4. The frames in Fig. 1.4 (a) to (d) are the frames from 50 to 65 of a pristine video from [145] with frame gap of 5 frames. The frames in Fig. 1.4 (e) to (h) are the corresponding frames from Fig. 1.4 (a) to (d) edited (upscale-cropped) using Adobe After Effects for simulating camera zooming. The frame gap is taken for visible understanding of zooming operation. The change in contents in successive frames of a video is very less for a frame rate of 25 frames per second (fps). Figure 1.4 (e) to (h) show that the edited frames appear as genuine as they are recorded as part of natural camera zoom. The frames in Fig. 1.5 (a) to (c) are the original frames from 42 to 102 of a pristine video having the visual of a two-storey building with frame gap of 40 frames from our dataset. Figure 1.5 (d) to (f) are the corresponding synthetically zoomed frames created using Adobe After Effects. The intended FOV of ground floor is completely removed by synthetic zooming. Hence, the activities happening in ground floor is completely missing in edited video. This helps a perpetrator in removing his/her presence without performing any state-of-the-art inter-frame or intra-frame video tampering.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 1.4: Sample frames from pristine and forged videos. (a) to (d) are the original frames from 50 to 65 of a pristine video with frame gap of 5 frames from [145]. (e) to (h) are their corresponding upscale-cropped frames created using Adobe After Effects.

1.3 Camera Tampering

Another concern in video surveillance systems is that most of them are left unattended due to limitations of manual monitoring [113]. Surveillance cameras will be installed in many places in an organization. The person-in-charge of surveillance system may not be able to view the visuals of all these recordings simultaneously at a time. Instead, he/she may be most interested in seeing only those which are having pertinent visuals from an organization's point of view. Manual monitoring is considered as wastage of time and human resource by some organizations. They leave surveillance systems unattended. Videos recorded by surveillance systems may be stored or backed up for some time and are periodically overwritten based on the storage constraints. If surveillance systems are left unattended, then recordings may get overwritten without even viewing them at least once. These recordings may suddenly find their relevance to identify and analyze an incident when an alternate mechanism reports a security incident. Perpetrators tendency to divert the attention of a person in charge of monitoring the recordings

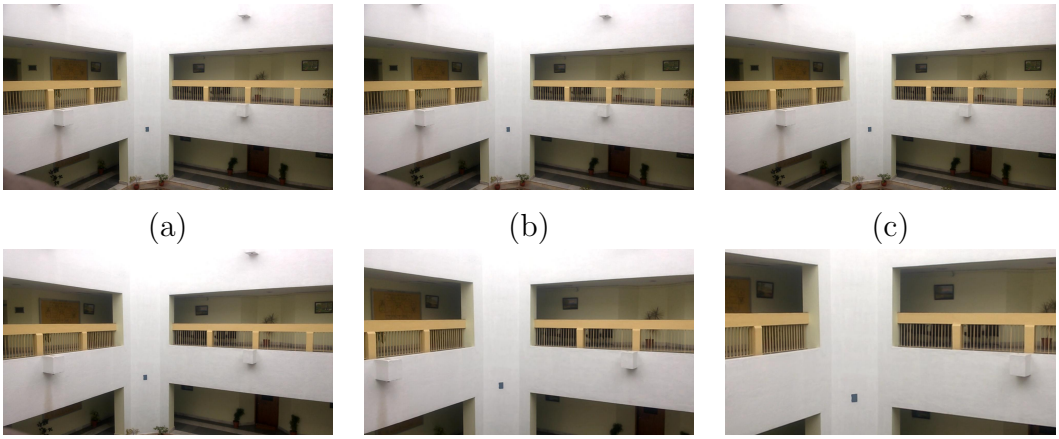


Figure 1.5: Sample frames from pristine and forged videos. (a) to (c) are the original frames from 42 to 102 of a pristine video having the visual of a two-storey building with frame gap of 40 frames from our dataset. (d) to (f) are their corresponding upscale-cropped frames created using Adobe After Effects. Intended FOV of ground floor is removed by synthetic zooming.

of a surveillance system is a significant concern over other practical difficulties of manual monitoring.

A video surveillance system should record videos with correct FOV. The quality of a recorded video should be good enough to support decision making. Or else, it may not be suitable for forensic investigation or other analysis purposes. To prevent capture of their images, criminals resort to tamper physical device itself by several techniques such as deliberately changing the camera view, covering the lens with a foreign object, spraying or defocusing the camera lens. Any action on video surveillance systems which interrupts the normal functioning of a surveillance camera can be treated as a camera tamper/sabotage event. These events can be unintentional or intentional. The events in unintentional category are those that occur because of environmental factors or on regular human interactions with the surveillance system. Environmental camera tampering include dust, fog, dirt on a camera lens, etc. Camera tampering because of human intervention include accidental contact with the camera while moving large objects through FOV, careless maintenance activity by cleaning staff, etc. The camera

tamper events in intentional category are those which are performed on purpose by a perpetrator which include:

- **Camera occlusion/covered:** Perpetrators obstruct the intended FOV of a surveillance camera. It is usually performed by placing an opaque object adjacent to the camera lens or by spraying paint on the camera. It is even possible with a piece of paper or cloth. Camera occlusion is of two types - partial occlusion and full occlusion. In an intended FOV, the amount of area obstructed by occlusion is the characteristic used for this classification.

- **Camera defocus/out-of-focus:** The intended FOV recorded during this camera tampering activity may not be of good quality. Perpetrators may change the focus settings of the camera so that the visuals recorded is unclear. Spraying water, oil or any viscous fluid on the camera will also give similar effects.

- **Camera displacement/moved:** Altering the position or direction of a surveillance camera so that it is not allowed to record the intended FOV completely. In a PTZ surveillance system, perpetrators may restrict the normal movement of a camera to limit its area of coverage so that the recording of entire FOV (actual pan area) is not possible.

A surveillance system is reliable if it can correctly detect and report camera tamper events in real-time.

Common video forensic methods (like video enhancement [45]) may fail in extracting features from foreground objects required for forensic analysis (like perpetrator's face) from the visuals recorded during camera tamper events. Video inpainting algorithms also fail in this scenario. These algorithms can not recreate the visuals of actual activities occurred at the crime scene and usually predict the values of those pixels which are to be reconstructed, from the information collected from past or future frames. Camera tamper events may negatively affect the video content analysis methods meant for safety and security [3, 16, 28, 57, 58, 62, 84, 90, 147]. Lavee et al. [57] discussed a video analysis tool to reduce user effort by identifying the events in a surveillance video. Ramstrand et al. [84] and Lee et al. [58] presented techniques for finding the heights of objects in

surveillance videos. Russo et al. [90] discusses a method for determining the size, shape, and location of objects in FOV. If a perpetrator lacks technical expertise in video editing and network intrusion, camera tampering is relatively simple than video tampering. It prevents the recording of his/her visuals in the crime scene without any technical expertise. Sometimes, a perpetrator has to gain remote or physical access to the storage media for editing the recorded video which is a challenging task. Video tampering detection methods [75, 101, 102, 105, 107] come into play only if a perpetrator is successful in video editing. Considering the difficulties in video tampering, perpetrators may prefer tampering the camera than video. This implies the relevance of an automatic mechanism for camera tampering/sabotage detection. If such a system exists which can trigger an alarm for suspicious camera tamper events viewed through the camera will be helpful to the general public.

Camera tampering detection methods may trigger false alarms due to various factors like illumination changes, crowd, large objects passing through and so on in the FOV. The reliability of the system depends on low false alarm rate. Reliability is a subjective aspect based on the operator's psychology as he/she may overlook future alarms assuming wrong ones. Countering the false alarms is a challenge which camera tampering detection algorithms fail to overcome in the literature.

In this thesis, videos complying with MPEG-4 and H.264 compression standards are taken for performance analysis of the proposed methods.

1.4 Problem Statement

Video tampering detection is a challenging task with the advent of sophisticated and affordable video editing tools. Passive video tampering detection methods are required for restoring the public trust towards video.

As video tampering requires technical expertise in video editing and network intrusion in some cases, perpetrators may prefer tampering the camera than video. Automatic camera tampering detection is a much-needed technique, if present may activate an alarm on camera tamper events for timely intervention.

From literature survey on video and camera tampering detection, we identified the following problems to work with for securing video surveillance systems:

- Real-time detection of camera sabotage with reduced false alarms and missed events
- Detection of Frame shuffling with duplication in static scene videos
- Tampering detection in videos having fixed and variable length GOP structure
- Minimize false positives due to zooming of lens in video tampering detection
- Differentiating synthetic zooming from natural for countering the anti-forensics on inter-frame video tampering detection
- Videos subjected to copy-move tampering using affine transformation have to be detected and regions of tampering have to be found.

1.5 Contributions

The contributions of this thesis can be divided into two categories: video tampering detection and camera tampering detection. An outline of our contributions is given in Fig. 1.6.

- **Contribution 1:** We propose a method to detect inter-frame video tampering using tamper traces from the spatio-temporal and compression domains. Our method is effective in detecting and differentiating various inter-frame video tampering such as frame insertion, deletion, shuffling, and duplication. It is tolerant to compression and capable of handling videos with fixed and adaptive GOP structure. The main contribution of our work is that we developed a method for frame shuffling detection and identifying the pristine frames involved in tampering. Also, a new method for zooming detection is proposed to decrease the false positives due to sudden zooming.

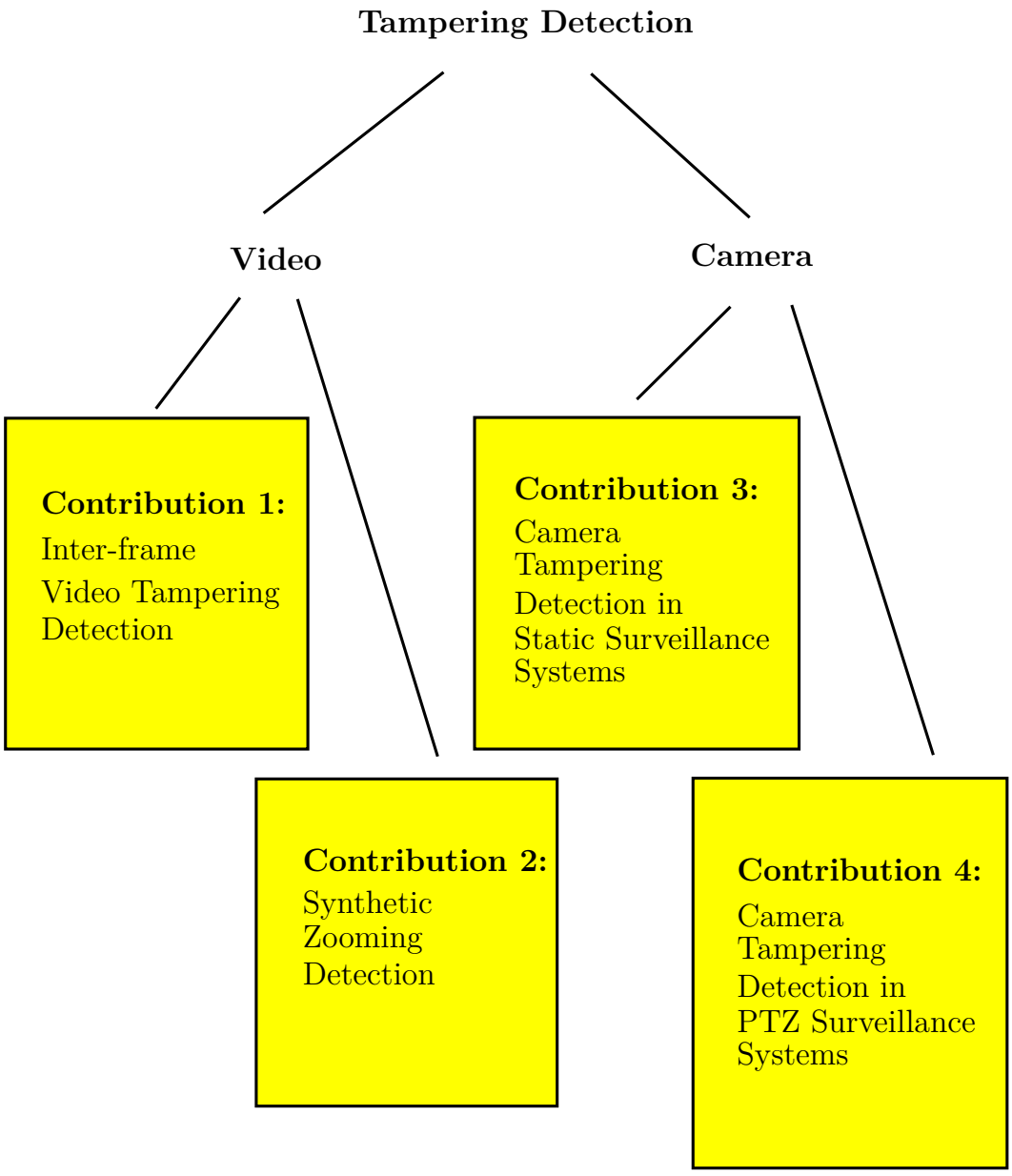


Figure 1.6: Outline of contributions of the thesis

– **Technique: Inter-frame video tampering detection**
Inconsistencies in velocity fields computed from successive frames are used as spatio-temporal artifacts, while, variations of prediction in the macro-block types of P-frame is used as a compression domain artifact.

Generalized extreme studentized deviate (ESD) algorithm [44] is used to find the abnormalities in velocity field sequence that occur as a result of inter-frame video tampering. The candidate locations of tampering identified using ESD are then inspected for compression artifacts. If it exists, videos are classified as tampered. We also propose methods to identify the type of inter-frame video tampering applied at these tamper locations.

  - **Technique: Zooming detection in video**
    The proposed method for zooming detection uses a key point based motion vector scheme. We extract Scale Invariant Feature Transform (SIFT) key points [68] from adjacent frames at tamper points. For finding the motion vectors between $(n+1)^{th}$ and n^{th} frames, we match the SIFT key points extracted from these frames. The motion vectors of a normal frame, when represented in a 2-D plane, appear scattered. Zoom-in and zoom-out operations during video recording lead to divergence and convergence of motion vectors respectively. To check divergence or convergence for zoom type classification, we compute the orientation of motion vectors.

- **Contribution 2:** We proposed a method for differentiating synthetic zooming from natural/optical camera zoom by identifying the location of zoom-out and zoom-in frames in a video.

  - **Technique: Synthetic zooming detection**
    To locate frames in between zoom-out and zoom-in operations, the zooming detection method proposed in Chapter 3 is used. Pixel variance correlation and Sensor Pattern Noise (SPN) variations of frames in between zoom-in and zoom-out operations are analyzed for determining tampering. Interpolation algorithms estimate pixel values at intermediate positions from its neighborhood. This process introduces dependencies between adjacent pixels. Pixel variance correlation enable us to identify these pixel dependencies. Interpolation process affects SPN distribution of resampled frames. The observations from

pixel variance correlation and SPN are obtained to decide if a video has undergone synthetic zooming or not.

- **Contribution 3:** We proposed methods for detecting camera tampering in static video surveillance systems. It can be implemented for real-time applications.
  - **Technique:** Foreground objects present in incoming frames are extracted using information from the background model. The size and static nature of a foreground object and edge details in incoming frames are the parameters used for deciding suspicious activity. An alarm is activated for persistent tamper events to reduce false alarm rate.
- **Contribution 4:** We proposed methods for camera tampering detection on panning surveillance cameras. The false alarm rate of the proposed system is less compared to the single existing method in the literature.
  - **Technique:** We create a mosaic view which is the background model corresponding to the intended FOV of the surveillance system. To identify the foreground objects in newer/current frames, we have to register and compare them to the background model created. To register current frames in background mosaic, SIFT key point [68] correspondences using homography are used. The area occupied by the foreground objects and their static nature, edge information and the number of frames recorded per camera pan cycle are the features used for detecting camera tamper events. An alarm is triggered on persistent tamper events to reduce false positives.

1.6 Organization of the thesis

The rest of the thesis is organized as follows. Chapter 2 discusses the basic concepts of video required for understanding this thesis. It provides an overview of the state-of-the-art methods in video and camera tampering detection. It also discusses anti-forensic techniques on video tampering detection. In Chapter

3, we describe the proposed methods for detecting inter-frame video tampering such as frame deletion, frame insertion, frame duplication and frame shuffling using artifacts of tampering from spatio-temporal and compression domains. A method for differentiating synthetic zoomed frames from optical camera zooming is presented in Chapter 4. In Chapter 5, we describe the proposed methods for detecting camera tampering events such as camera defocus, displacement, and occlusion in static video surveillance cameras. Camera tampering detection in PTZ surveillance system is presented in Chapter 6. Chapter 7 summarizes the contributions of the thesis and outlines the future research directions.

Chapter 2

Review of Related Works

Video tampering attacks can be performed in spatial, temporal and spatio-temporal domains. Region splicing and copy-paste tampering occurs in spatial and spatio-temporal domains. Frame insertion, frame deletion, frame duplication and frame shuffling occurs in temporal domain. Video can be thought of as a sequence of images called frames, displayed over a period of time. The methods for image forgery detection [59, 73, 82, 127, 150] may be employed at frame level. But time domain, the third dimension of a video has significant role in video compression. It may introduce compression noise and motion artifacts depending on the codecs used for compressing video. The application of image tampering detection techniques as such to videos may fail to identify the artifacts of video tampering leading to misclassification. Also, exclusion of information from the temporal domain may increase computation cost.

Video tampering detection techniques can be classified into passive (also called as blind) and active as discussed in Section 1.1 of Chapter 1. Any editing operations will leave some footprints in video which could be leveraged for checking the authenticity of video in passive video forensics. These artifacts consist of increase in prediction error; abnormalities in noise, motion vectors and motion residues; high temporal and spatial correlation between frame intensity values, Variation of Prediction Footprint (VPF), inconsistencies in optical flow, Motion-Compensated Edge Artifact (MCEA), quality of frames, and so on. A perpetrator who has adequate knowledge of video forensic methods may use anti-forensics to conceal the tampering activity. The application of anti-forensics reduces the effect of the

artifacts left by video editing process. However, it leads to the inclusion of new artifacts.

This chapter provides an overview of the state-of-the art methods in video and camera tampering detection. Anti-forensic techniques are also discussed. We categorize various types of tampering and their detection techniques. Image resampling detection methods are also discussed which we studied for synthetic zooming detection in video due to the lack of literature in the concerned area. We also try to figure out the drawbacks of each of them on a broader canvas.

The rest of the chapter is organized as follows. Section 2.1 deals with the background concepts which is required for understanding this survey. Section 2.2 gives information on video editing software. Section 2.3 is dedicated to passive video tampering detection methods. Image resampling detection methods are discussed in section 2.4. Section 2.5 deals with anti-forensic techniques in the literature of video tampering detection. Section 2.6 discusses camera tampering detection methods. Section 2.7 deals with discussion regarding video and camera tampering datasets available as well as created by us. The performance measures used for evaluating the proposed methods for video and camera tampering detection are given in Section 2.8. Section 2.9 discusses the limitations or challenges faced by most of the methods in the literature.

2.1 Video Basics

A video is a sequence of images displayed continuously for conceiving motion perception by exploiting the persistence of vision of the human visual system. A recorded video may have audio as well as the visual content of the scene captured. Our work focuses only on the visual data component of video files. A raw video consumes large storage space in general; therefore, acquisition devices use compression algorithms to solve this issue. Motion JPEG, MPEG-1 [96], MPEG-2 [96], MPEG-4 [88], H.264 [88], etc. are some of such video coding standards. Video has three dimensions - 2 spatial and 1 temporal as shown in Fig. 2.1.

Frame size in pixels or resolution is defined as number of horizontal pixels $\times$ number of vertical pixels, for example 1280 $\times$ 720 or 1920 $\times$ 1080. Table 2.1 provides summary of various video resolution standards. Scanning system is

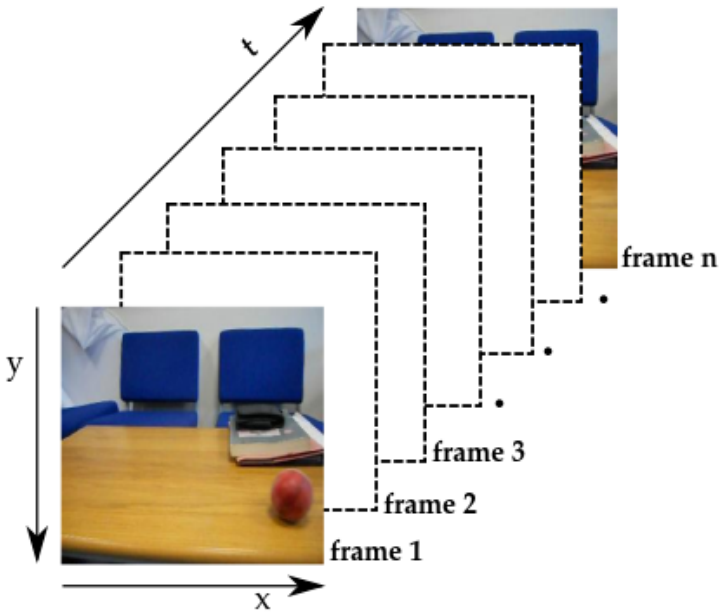


Figure 2.1: Video as a sequence of frames

identified with the letter p for progressive scanning or i for interlaced scanning. Often the number of horizontal pixels is implied from context and is omitted. In Table 2.1, 1280 × 720 and 1920 × 1080 is represented as 720p and 1080p respectively. CGA/EGA stands for Color or Enhanced Graphics Adapter, CIF stands for Common Interchange Format, VGA stands for Video Graphics Array, HD stands for High Definition, FWXGA stands for Full Wide Extended Graphics Array and UHD stands for Ultra High Definition. We have used CGA/EGA, CIF, VGA, Misc VGA and HD in 720p and 1080p resolution videos for testing our proposed methods for video and camera tampering detection.

Spatial and temporal redundancy are the key factors that aid in accomplishing compression. Spatial redundancy deals with transform domain coding while temporal redundancy deals with predictive coding. The variation between two consecutive frames in a video is usually negligible. Therefore, storage of every frame in a video leads to redundancy. The compression methods keep reference frames and predict the remaining frames from them instead of storing all the

Table 2.1: Summary of video resolution standards

Sl.No	Standard	Resolution (dots × lines)	Pixels
1	CGA/EGA	320 × 240	76,800
2	CIF	352 × 288	101,376
3	VGA	640 × 480	307,200
4	Misc VGA	720 × 576	414,720
5	720p (HD)	1280 × 720	921,600
6	FWXGA	1366 × 768	1,049,088
7	1080i, 1080p (HD)	1920 × 1080	2,073,600
8	4K (UHD)	3840 × 2160	8,294,400
9	8K (UHD)	7680 × 4320	33,177,600
10	16K (UHD)	15360 × 8640	132,710,400

frames. Based on compressed representation of the video, the frames are categorized into three: (1) I-frame (Intra-coded), (2) P-frame (Predicted) and (3) B-frame (Bi-directionally predicted). I-frames exploits spatial redundancy. It is handled more like an image without a reference for encoding. The initial frame of a video is an I-frame. P-frames predict its value from prior I or P frames. P-frames saves only the variations from its reference, thereby becoming more efficient than I-frames. B-frames predict its value from previous and following reference frames. Hence, P and B frames employ temporal redundancy whenever feasible in conjunction with spatial redundancy. B-frames give increased compression than P-frames. It is always impossible to predict the complete frames in a video from the initial frame (e.g., videos with constant changing background). I-frames require to be inserted at periodic intervals or depending on the motion in the video. The video is split into fragments (a group of frames), called Group of Pictures (GOP). Each GOP has a distinct structure with I-frame as the first frame followed by B and P frames.

Common encoders employ constant number of frames in a GOP (Fig. 2.2) for easy implementation but at the cost of video quality and compression ratio.

H.264 video encoder supports both fixed and adaptive GOP (AGOP) structure. In an AGOP, frame count within a GOP varies. This count can increase up to 250 frames based on the nature of the video content. A pictorial illustration of AGOP is shown in Fig. 2.3. It contains a very few frames in a GOP in case of a dynamic video as the background and content change rapidly. Therefore, better compression and visual quality are assured. The frames are subsequently split into macro-blocks (MB). If a frame is in RGB, then it will be converted to YCbCr or YUV to achieve compression in 4:2:0 format. The MB representing a 16×16 pixel block consists of four 8×8 sample blocks from Y component and one 8×8 sample block from each chrominance components (sensitivity of the human visual system is more towards luminance component) yielding a total of six 8×8 blocks in an MB. The macro-block encoding in P or B frames is performed by examining the reference frames for the best matching MB. The location and displacement (motion vectors) of the matched MB is saved to produce the predicted frame. The predicted frames are compared with original frames to obtain the error residues that are coded in a JPEG-like process. These error residuals and motion vectors (MV) are used for encoding each MB. The 3 main types of MBs are Intra-coded (I-MB), Predicted (P-MB), and skipped (S-MB). There is no coding needed for S-MBs. They are directly copied from other frames and require no MVs. I-MB contains MBs encoded without using temporal prediction while P-MBs use temporal prediction. A frame is coded as one or more slices containing one or more MBs.

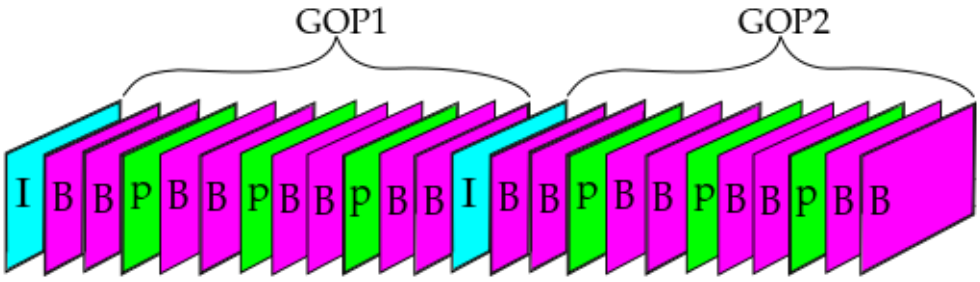


Figure 2.2: Structure of Fixed GOP

With this brief introduction on video, we advance to video forensics. The classification of digital video forensics is presented in Fig. 2.4. “Illegal Reproduction

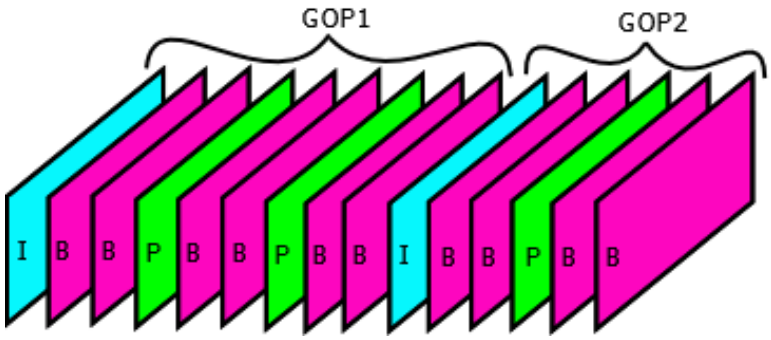


Figure 2.3: Structure of Adaptive GOP

of Videos", the first block in Fig. 2.4 focuses on recognizing fraudulent copies of the video for the protection of copyright. The second block differentiates real videos from computer-generated videos. "Tampering detection", the third block, can be defined as the detection of malicious alteration of video content. "Source camera identification" deals with the identification of a particular device used for recording the video under examination. "Video Steganography" is the art of stealthily hiding data in a cover media (in this case, video). Video steganalysis is the detection of this secret information in a video file. Milani et al. [75] and Rocha et al. [89] presented a summary of video forensics. A few methods ([19, 24, 56, 63, 111]) in the literature succeed in localizing regions of tampering in a video while the others ([10, 13, 20, 105, 110, 131]) can only classify it as tampered or not.

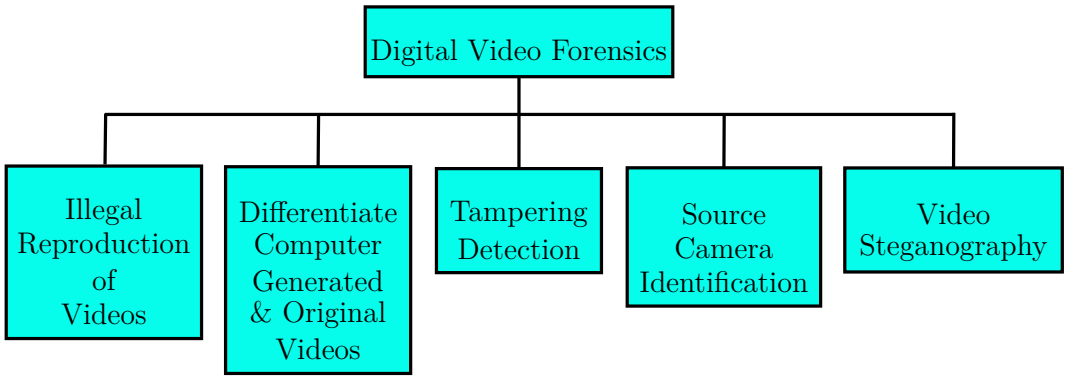


Figure 2.4: Classification of Digital Video Forensics

2.2 Tools for Video Editing

The advent of low-priced and easy-to-use video acquisition devices (e.g., smartphones, digital cameras, webcams, camcoders, etc) and video editing software (e.g., FFmpeg, Mocha Pro, Adobe Premiere Pro, CyberLink's PowerDirector) have made recording and altering the contents of the video possible for literally anyone.

2.2.1 FFmpeg

FFmpeg is a leading multimedia framework which can perform decoding, encoding, transcoding, mux, demux, streaming and filtering. It can support almost every file formats. It offers only a command-line interface for the processing of video and audio files. It can be used for basic editing like trimming and concatenation, scaling of video, post-production effects and standards compliance.

2.2.2 Adobe After Effects

Adobe After Effects is a digital visual effects, motion graphics, and compositing software. It is widely used for compositing and animation. It also functions as non-linear editor and media transcoder.

2.2.3 Mocha Pro

It is a set of tools developed by Imagineer Systems Limited, for editors and artists to save time on complex visual effects tasks including planar motion tracking, object removal, and lens distortion. It can be used as a plug-in for other video editing software.

2.3 Video Tampering Detection

We explore passive video tampering detection methods shown in Fig.2.8 (enclosed in dotted box) based on the type of tampering they address:

1. Double or multiple compression detection,

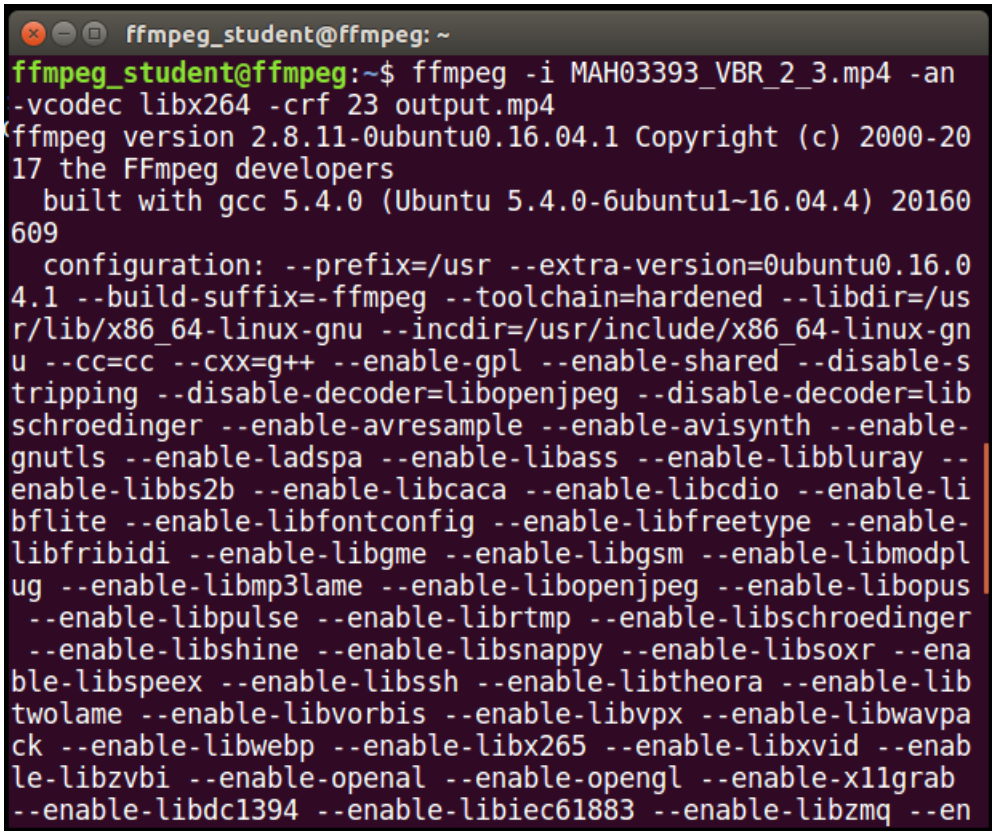


Figure 2.5: Command-line interface of FFmpeg running on linux platform

2. Region tampering detection

3. Inter-frame video tampering detection

Each of these are explained in detail in the following subsections. The summary and limitations of the features used by the methods in these categories are given in Table 2.2, 2.3 and 2.4. The survey papers by Wahab et al. [129] and Sitara and Mehtre [101] give idea on passive video tampering detection techniques. Joshi and Jain [48] and Singh and Aggarwal [98] brought out review papers on video authentication techniques.

2.3.1 Double or Multiple Compression Detection

A compressed video has to be decompressed in the first place to perform tampering activity. Once this activity is completed, the tampered video may be saved

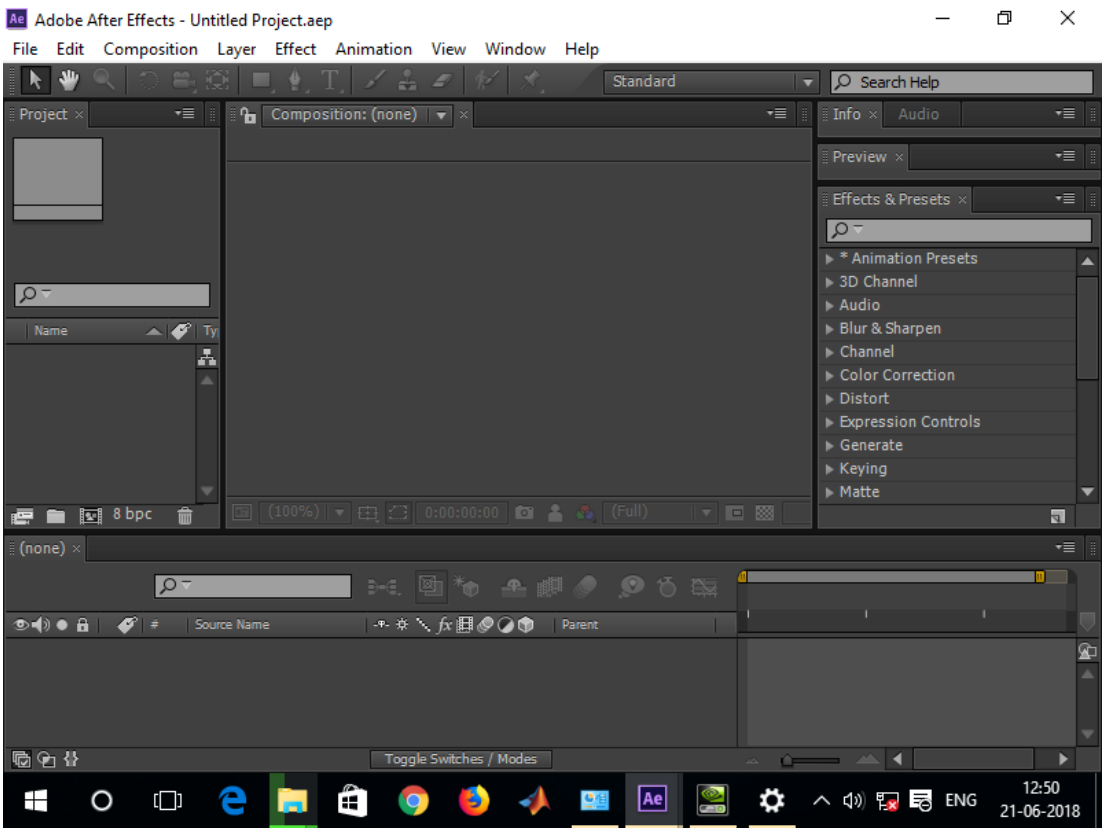


Figure 2.6: Adobe After Effects interface running on Windows OS. The screenshot illustrates how a video is cut using markers on the timeline of a video.

back in the compressed format. This follows that, if a video has undergone double or more compressions, then it may be a forged one. Such multiple recompressions will leave specific footprints or artifacts in the video, backing tampering detection. The reason for the development of these artifacts is that the coding parameters used in first and subsequent compressions will be different in most cases. In a pristine video, the distribution of block Discrete Cosine Transform (DCT) coefficients (DC/AC) may be Gaussian or Laplacian. The quantized DCT coefficients after quantization will also follow this. However, if recompression uses different encoding parameters, then it may not follow this distribution. A summary of techniques that are discussed in this section is summarized in Table. 2.2.

Wang and Farid [131] developed a method for MPEG video tamper detection using concepts of double JPEG compression and motion vectors. When an MPEG

Figure 2.7: Screenshot of Mocha Pro being launched with it's splash screen

video is recompressed, I-frames undergo double JPEG compression. Also, frame insertion or deletion causes frames in different GOPs in the original video come together in the recompressed video which increases the motion estimation errors. In a fixed GOP video, it contributes periodic spikes in the Discrete Fourier Transform (DFT) of P-frame prediction errors. This method cannot be applied on bulk videos as it needs human inspection which is liable to human error. In [134], the distribution of DCT coefficients of MBs in I-frames is evaluated for comparison with an expected distribution. A variant of normalized Euclidean distance is employed for measuring the deviation. An MB is categorized as recompressed or not based on the probability computed from this distance.

Other works based on DCT coefficient distribution in I, P and B frames are discussed in [108, 139, 140]. An SVM classifier with feature vector consisting of 3 features: root mean squared error (RMSE), sum of squared errors (SSE), and R-Square is presented in [139]. In [114], the method used is similar to that in [139] and [14]. A 12-D feature is created by fitting the distribution of 1^{st} digits

of Alternating Current (AC) coefficients from each I-frame, using parametric logarithmic law. Milani et al. [74] presented a method for recovering the number of compressions employed on a video using multiple SVM leveraging the Benford's law. Along with double compression detection, Vazquez-Padin et al. [125] discussed a method for estimating the GOP size used in first compression. The basic idea behind this work is explained in Subsection 3.1.1 of Chapter 3. As they employed VPF, we can expect better performance in H.264 videos.

Table 2.2: Summary of Double or Multiple compression detection techniques

Features Used	**Ref.**	**Limitations**
Statistical patterns in the distribution of DCT coefficients	[131]	Works only on fixed GOP; sensitive to noise and changes in GOP; fails when frames involved in tampering are integral multiples of GOP size
	[134]	Accuracy decreases when the ratio between the 1^{st} and 2^{nd} quantization scale (qs) is $<$ 1.7; analysis is done with I-frames only
	[108]	Fails in slow motion videos; accuracy decreases when the bitrate of 2^{nd} encoder is less than that of 1^{st} encoder; analysis is done with I-frames only
	[139]	Performance decreases with increase in target output (transcoding) bitrate
Statistical patterns in first digit distribution of DCT coefficients	[114]	Analysis is done with I-frames only
	[74]	Performance decreases with multiple compression
VPF	[125]	Works only on fixed GOP; fails when G1=G2; accuracy drops with increase in G1; where G1 and G2 are the GOP sizes used in 1^{st} and 2^{nd} compression respectively
Markov statistics of quantized DCT coefficients	[47]	Fails if 2^{nd} qs is same as 1^{st}; performance degrades when 2^{nd} qs is an odd multiple of 1^{st}

Table 2.2 Summary of Double or Multiple compression detection techniques (Continued):

Features Used	**Ref.**	**Limitations**
Markov statistics of compression noise	[86]	Fails if 2^{nd} qs is same as 1^{st}
Pixel prediction in GOP	[112]	Works only on fixed GOP; accuracy decreases when the ratio between the 1^{st} and 2^{nd} qs < 1.3; works on static camera videos only
Blocking artifact	[69]	Fails when frames involved in tampering are integral multiples of GOP size; works only on fixed GOP
Correlation of a video with its recompressed version where the codec and coding parameters are the same	[7]	Performance degrades if coarse quantization is adopted in 2^{nd} encoding; content dependent
Number of different coefficients between singly and doubly compressed I-frames, number of different coefficients between doubly and triply compressed I-frames	[40]	Works only on those videos where the bitrate of 1^{st} and 2^{nd} compression are same; performance depends on the proper selection of recompression bitrate

Jiang et al. [47] presented a recompression detection method using Transition Probability Matrix (TPM) from 1^{st} order Markov statistics of differences along vertical, horizontal, minor diagonal and major diagonal directions of quantized DCT coefficients. Fisher's linear discriminant (FLD) analysis is used for classifying GOPs as singly or doubly compressed. Ravi et al. [86] computed compression noise based on Huber Markov Random Field (HMRF) and maximum a posteriori (MAP) criteria. TPM is computed from 1^{st} order Markov statistics of differences along eight directions in 16×16 block (8-connected neighborhood). A thresholding approach based on the number of frames being classified as forged is used for categorizing a video clip as forged.

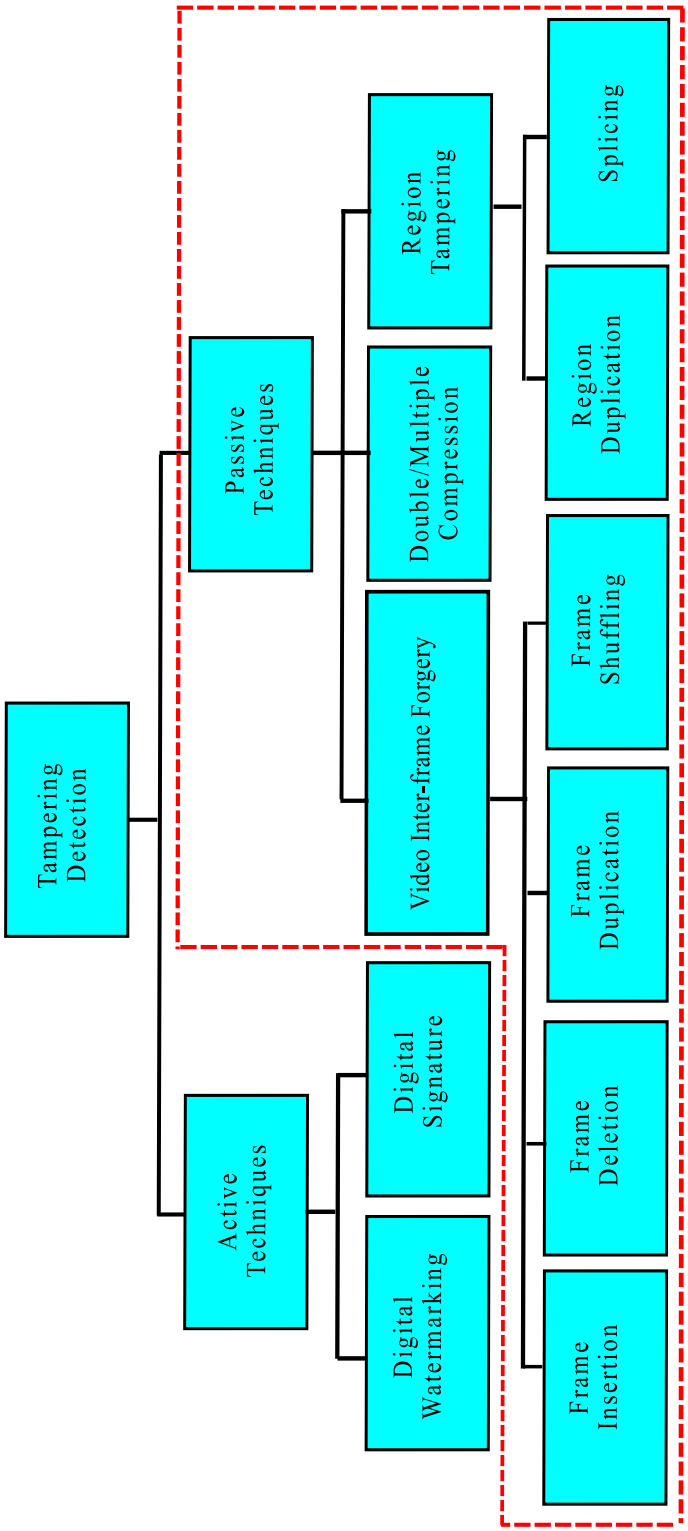


Figure 2.8: Classification of Tampering Detection

Subramanyam and Emmanuel [112] classified frames in a GOP as double compressed based on pixel prediction from spatially co-located pixels from the rest of the frames in a GOP. A thresholding approach is then applied on the error computed between true and predicted value. Luo et al. [69] discussed recompression detection using blocking artifacts of compression. Bestagini et al. [7] presented a method for identifying the codec type used in the first compression. When a reconstructed sequence is re-encoded with same codec and coding parameters used in previous compression, it yields a sequence which is highly correlated with the input compressed video. Bestagini et al. [7] utilized this concept by compressing the reconstructed sequence from input video using several codecs and coding parameters inspecting for similarities between them.

Most of the algorithms for recompression detection fails when the video under investigation is recompressed with same coding parameters used in the first compression. Huang et al. [40] presented a method for handling this scenario based on the assumption that the number of distinct DCT coefficients in two subsequent compressed versions of a video decrease monotonically. That is, the number of distinct coefficients in I-frames that underwent single and double MPEG-2 compression with the same bitrate will be substantially high than that which underwent double and triple MPEG-2 compression with the same bitrate.

A method for finding the GOP structure based on noise variance behavior is presented in [141]. It is useful for tampering detection in videos with unknown GOP structure. I-frames can be differentiated easily as its noise power is either low or high. The period of peaks in autocorrelation sequence provides GOP size. It works better on static scene videos. The performance degrades in dynamic scene videos due to the difference in noise power in different scenes. A method for finding the quantization parameters in H.264 videos using motion residuals is discussed in [116]. It is applicable on fixed GOP videos. Li and Forchhammer [60] proposed a method for estimating the compression parameters from MPEG-2 videos without accessing its original. Valenzise et al. [124] computed the quantization parameters and MVs from H.264 videos without using the encoded bitstream. The use of de-blocking filters in H.264 videos degrades its performance.

Tamper artifacts in the prediction residual sequence computed from MVs of the foreground objects are presented in [36]. It is developed for static background

videos.

2.3.2 Region Tampering Detection

The methods that are capable of localizing the regions of tampering in the spatial as well as temporal domain such as - region duplication/copy-paste and splicing are discussed here. In region duplication, small portions (blocks or regions) of frames from a sequence are copied and pasted at another frame position in the same video. A pictorial representation of region duplication is shown in Fig. 2.9(a). The 3D-planes in figures correspond to frame sequence where x and y denote frame size and t is the duration of video. Region duplication tampering is indicated with blue coloured frame regions which are copied from the video and placed at a later frame sub-sequence in the same video. Figure 2.10 shows an example of copy-paste tampering from SULFA dataset [83] (discussed in Section 2.7.1). Figure 2.10 (a) - (e) correspond to frames from 101 - 105 of a pristine video. Figure 2.10 (f) - (j) are the frames coming at corresponding positions in the tampered video. The lady in the recorded scene is erased by copy-paste forgery. In splicing, frame regions or objects taken from one video are inserted into another video. Region splicing is indicated with brown coloured frame regions in Fig. 2.9(b) which correspond to the inserted frame regions from a different video. Retouching of tampered regions or application of affine transformations make video tampering detection strenuous. A summary of techniques discussed in this section is given in Table.2.3.

Wang and Farid [133] proposed methods for video tampering detection in de-interlaced and interlaced videos. Video tampering disturbs the spatial and temporal correlations as well as motion across fields in de-interlaced videos. In interlaced videos, it disturbs the equality of motion between the fields of a frame. They proposed a method using correlation for detecting the duplication of frames or regions in [132]. To detect video tampering performed in videos recorded using a panning surveillance camera, the motion and period of the camera are taken into account. Hsu et al. [37] presented a method for tamper detection which models the distribution of block-based correlation of noise residuals in Gaussian

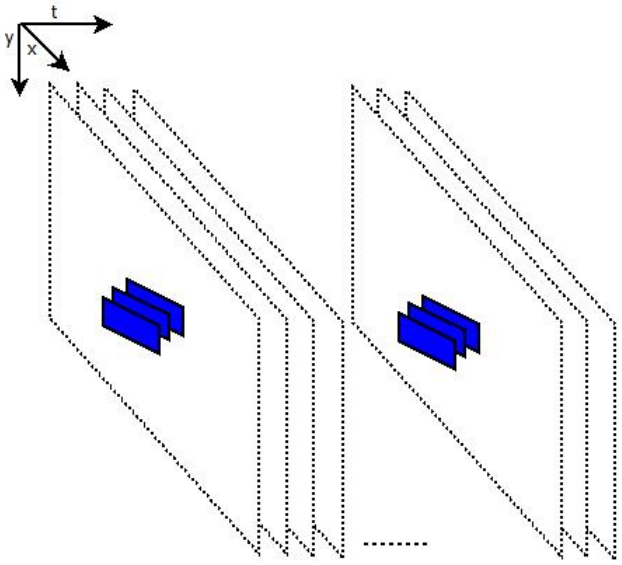
mixture model (GMM). Expectation-Maximization (EM) is then employed for estimating the parameters of GMM.

Li et al. [61] used the orientation and magnitude of MVs obtained from consecutive frames for differentiating authentic and forged regions. These characteristics of MVs are usually uniform in pristine videos with normal movement compared to the tampered region. A method to detect region tampered frames (add/remove objects) using features extracted from motion residuals of each frame is presented in [13]. To handle AGOP videos, motion residues are generated using collusion operators. Feature extractors in image steganalysis: SPAM, CC-JRM, CF*, J+SRM, SRM, CC-PEV, and CDF are utilized for extracting the forensic features from motion residuals [1].

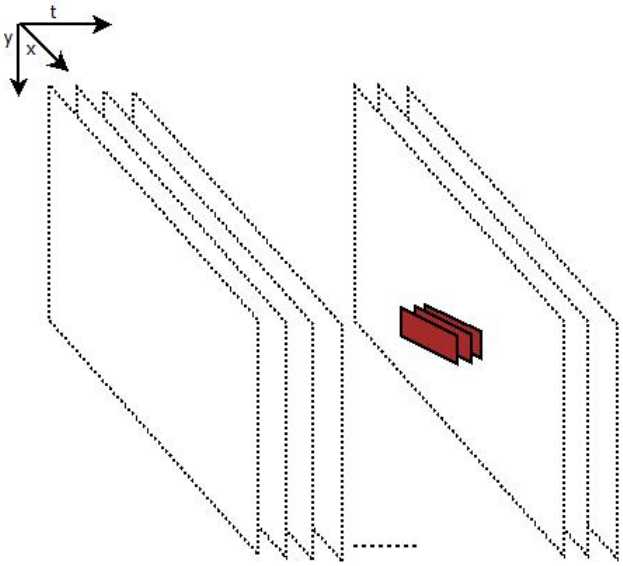
Usually, removal of moving objects by video inpainting contributes ghost shadow artifacts. Zhang et al. [146] detected this artifact using the inconsistencies in active foregrounds and its motion. Another approach for detecting video inpainting is presented in [63]. Frame groups in grayscale are analyzed for spatio-temporal coherence independently that yields group coherence abnormality pattern (GCAP). It is used for locating the regions with abnormal coherence. A method for detecting moving foreground removal in static videos employing the features extracted using k-Singular Value Decomposition (k-SVD) from the difference image obtained between non-tampered reference and current frames is presented in [106].

Kobayashi et al. [54] presented a noise level function (NLF) based method for detecting tampered regions in static background videos. Spliced regions are identified using MAP estimation on NLF where the noise characteristics of these regions are irregular compared to the rest of the video. Chetty et al. [15, 32] discussed tamper detection techniques for low-bandwidth videos using noise and quantization residues from inter-frame and intra-frame pixel sub-blocks. These features are fused after transforming to the cross-modal subspace using Canonical Correlation Analysis (CCA), Cross-modal Factor Analysis (CFA), and Latent Semantic Analysis (LSA).

[1] Abbreviations of these feature extractors are taken from http://dde.binghamton.edu/download/feature_extractors/



(a)



(b)

Figure 2.9: Pictorial representation of video intra-frame forgery. (a) Region duplication - indicated in blue are small frame regions copied from a continuous sequence of frames in a video and pasted at later spatio-temporal positions in the same video; (b) Region splicing - indicated in brown are the frame regions inserted from a different video

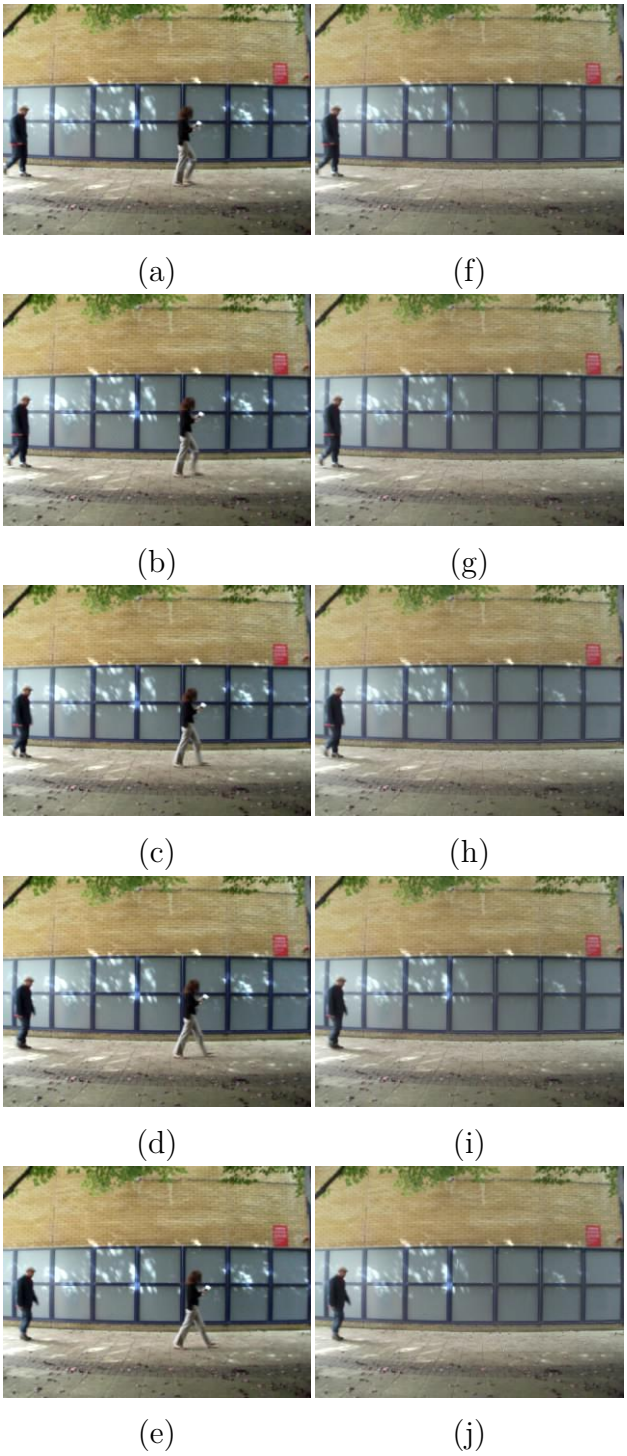


Figure 2.10: Region duplication/Copy-paste tampering from SULFA dataset [83]. (a - e) correspond to frames from 101 - 105 of a pristine video. (f - j) are the frames coming at corresponding positions in the tampered video. Lady in the scene is erased by copy-paste forgery.

Table 2.3: Summary of Region tampering detection techniques

Features Used	**Ref.**	**Limitations**
Special & temporal correlation	[133] (Deinterlaced)	Compression artifacts and noise will degrade performance
	[132]	False positives on uniform areas like clouds; fails when duplication is performed synchronously with the period of the moving camera; fails to detect duplication in static background video frames
Motion vector based	[133] (Interlaced)	Compression artifacts and noise degrades performance
	[61]	Works well on static background videos compared to dynamic background; Compression degrades performance; may fail if tampering is done properly
Temporal noise	[37]	Sensitive to quantization noise, too high or too low illumination; content dependent; works well on static background videos only
	[54]	Deals with static scene videos only; performance depends on the codec used for video compression; post-processing operations like tuning brightness and contrast affects noise characteristics heavily
	[78]	Detection accuracy decreases with increase in compression ratio
Motion residuals	[13]	Extact localization of forged objects is not possible. Bit rate reduction degrades performance.
Ghost shadow artifact	[146]	Tampered area localization is not accurate; works well in static background videos only
Noise and quantization residue	[15, 32]	Sensitive to noise, too high or too low illumination; works well on static background videos only
HOG	[111]	Works only on fixed GOP; copy-paste tampering alone is addressed

Table 2.3 Summary of Region tampering detection techniques (Continued):

Features Used	**Ref.**	**Limitations**
VPF, histogram of DCT coefficients	[56]	Presence of B-frames are not considered; works with Variable Bit Rate (VBR) coding only
Spatio-temporal coherence	[63]	Performance decreases with increase in compression
Difference between current & non-tampered reference frame	[106]	Works with static background videos; detection accuracy decreases in 2 cases: removal of small foregrounds, or fast moving foreground
Zernike moments and 3D patch match	[19]	Accuracy is very low
Optical flow	[10]	Performance depends on ROI mask selection which is a manual process; works only on fixed GOP; fails when displacement of forged parts is not a multiplier of GOP's length and performance decreases in videos with high motion content.

Copy-paste tampering detection using Histogram of Oriented Gradients (HOG) is presented in [111]. Block based HOG feature comparison of all blocks in a frame is used for determining intra-frame copy-paste. Block based HOG feature comparison between the blocks of one image to that of other frames in a GOP is used for determining inter-frame copy-paste forgery. Labartino et al. [56] utilized the method in [125] for estimating GOP size of first compression for locating the double intra-coded frames and performing double quantization analysis on them. Pandey et al. [78] proposed SIFT key point and k-Nearest Neighbors (k-NN) based matching method for copy-paste detection in spatial domain and noise residue cross correlation method in temporal domain. Amiano et al. [19] extended 2D patch match method for copy-paste tampering detection in images [17] to videos by incorporating the information from temporal domain using Zernike moments and 3D patch match.

Copy-paste tampering detection using Lucas Kanade optical flow between

consecutive frames is presented in [10]. A frame is divided into: Suspicious region and remaining areas. Suspicious region denote the ROI of attackers. Optical Flow (OF) and Optical Flow Variation Factor (OPVF) of these Suspicious ROI and other areas are estimated separately. Based on the periodic peaks in the autocorrelation sequences obtained from these two parts, a video is classified as tampered ot not.

2.3.3 Inter-frame Video Tampering Detection

Inter-frame video tampering detection deals with: (1) frame insertion, (2) frame deletion/removal, (3) frame duplication, and (4) frame shuffling. These tampering types are discussed in Section 1.1 of Chapter 1. A summary of the techniques discussed in this section is presented in Table.2.4.

Table 2.4: Summary of Inter-frame Video tampering detection techniques

Tamper Type	**Features Used**	**Ref.**	**Limitations**
Frame deletion	MCEA in P-frames	[109]	Works only on fixed GOP containing at least 3 P-frames; Fails when frames involved in tampering are integral multiples of GOP size; fails when the impact factors of tampered and pristine videos comes in the same interval; performance degrades in slow motion videos
		[20]	Works only on fixed GOP; fails when frames involved in tampering are integral multiples of GOP size
	Periodic artifacts in DCT coefficients of B and P frames	[110]	Fails when frames involved in tampering are integral multiples of GOP size
	SARP	[66]	Works only on fixed GOP and fails when frames involved in tampering are integral multiples of GOP size

Table 2.4 Summary of Inter-frame Video tampering detection techniques (Continued):

Tamper Type	Features Used	Ref.	Limitations
	Differences of mean motion residual in consecutive frames	[24]	Increase in bitrate decreases accuracy; poor performance on very slow motion videos
	Prediction residuals, percentage of I-MBs, quantization scales and reconstruction quality	[94]	Fails when frames involved in tampering are integral multiples of GOP size
	Variation in prediction residuals and I-MBs	[144]	Fails when frames involved in tampering are integral multiples of GOP size
Frame interpolation	MVs, periodicity of squared prediction error	[8]	Compression affects performance; fails to detect downsampling where interpolation factor >= 2
Frame insertion	Multi-Level Subtraction	[38]	Sensitive to compression
Frame duplication	CHD	[65]	Fails when duplicated frames are shuffled
	Correlation of SVD features of frame sequences	[142]	Fails when frames involved in forgery is less than the window size considered and when duplicated frames are shuffled
	Correlation between suspicious frames	[99]	Compression decreases performance; fails when duplicated frames are shuffled
Frame deletion & duplication	Consistency of VFI	[138]	Frame duplication detection accuracy decreases with increase in compression
Frame repetition & deletion	Motion energy at SROI, average object area and entropy	[34]	Better performance on high motion content videos only

Table 2.4 Summary of Inter-frame Video tampering detection techniques (Continued):

Tamper Type	**Features Used**	**Ref.**	**Limitations**
Frame insertion & deletion	P-frame prediction error sequence	[49, 105]	Sensitive to noise; [49] fails when frames involved in tampering are integral multiples of GOP size
	Differences of correlation coefficients of gray values between consecutive frames	[130]	Increase in compression decreases performance; frame deletion accuracy is less
	VPF	[31]	Works only on fixed GOP; fails when G1=G2; localization of tamper points is not accurate
	QCCoLBP	[148]	Cannot differentiate frame insertion and deletion; performance decreases when less number of frames are deleted
	Optical flow	[12]	Works well on frame insertion when inserted frame have a different background scene
	BBVD	[151]	Detection accuracy decreases when number of frames inserted or deleted < 25; work on static camera videos and forged videos having only one type of forgery where insertion or deletion is performed once
	Sum of absolute differences between video frames before and after applying deblocking filter	[107]	Deblocking filter and intra prediction in H.264/AVC degrade the performance; fails when same rate control method is used to transcode the video

Table 2.4 Summary of Inter-frame Video tampering detection techniques (Continued):

Tamper Type	**Features Used**	**Ref.**	**Limitations**
Fixed Frame repetition & spatio-temporal region duplication	Residual from consecutive frames, cross-correlation of residual	[9]	Detection accuracy reduces with increase in compression
Frame insertion, deletion & duplication	Optical flow	[135]	Frame deletion detection accuracy is less; frequent motions, compression, and complicated backgrounds degrades performance
	ZOCM	[67]	Fails in dynamic background videos
	Prediction residual and optical flow	[50]	Fails when frames involved in tampering are integral multiples of GOP size

Su et al. [109] detected frame deletion using MCEA which arises from motion-compensated prediction and blocking impairment. As the distance between I-frame and another frame within a GOP increases, it introduces artifacts at block boundaries of the same frame. Frame deletion may decrease the temporal correlation between neighboring frames at frame deletion point which in turn introduces artifacts at block boundaries. This disturbs MCEA energy. The MCEA distribution computed from the P-frames present in a GOP is used for classifying a video clip as tampered or not.

Lin et al. [65] divided the video under investigation into sub-sequences. The similarities between the consecutive frames in candidate and query sub-sequences are estimated using color histogram difference (CHD). If the CHD between candidate and query subsequences are highly correlated, then the spatial correlation of CHDs obtained block-wise between the frames in query and candidate clips are computed which are then thresholded for frame duplication detection. Su et al. [110] used the periodic artifacts in the DCT coefficients of B and P frames

for frame deletion detection in MPEG videos. Dong et al. [20] extended the work presented in [109] using Fast Fourier Transform (FFT) on MCEA difference. For a GOP in the pristine video, the difference of MCEA (dM) between consecutive P frames is comparatively steady. Inter-frame video tampering followed by recompression increases motion compensation errors in P frame which in turn affects dM and contributes periodic spikes in the FFT of dM. A method for detecting frame deletion and addition in fixed and AGOP videos is discussed in [105]. The periodic increase in prediction error is leveraged for tampering detection in fixed GOP videos whereas an energy detector is used in AGOP videos. A method for detecting the anti-forensics discussed in [104] is also presented in [105]. The unusual MVs with 0s are identified by comparing the MVs of a video under investigation to the MVs estimated from the same video.

Bestagini et al. [9] discussed methods for detecting two cases of spatio-temporal region tampering: (1) a video sub-sequence replaced with fixed image and (2) a video sub-sequence replaced with another sub-sequence of the same video. In the former, the residual between consecutive frames (exact 0) is used. In the latter, the cross-correlation of small 3-D residual blocks is used. In [8], a method to detect temporal interpolation is presented. If the frame rate of the video sequences used in temporal splicing are different, then these videos need to be temporally interpolated beforehand. Motion compensated interpolators with minimized visual artifacts are used. MVs computed from consecutive frames are used for tampering detection. The interpolation factor is estimated from the periodicity present in the FFT of squared prediction error.

Sequence of Average Residual of P-frames (SARP) based method for frame deletion detection in H.264 videos is presented in [66]. The SARP of those P-frames which are shifted to a new GOP in second compression after frame deletion is high compared to that computed from the pristine P-frames in the same GOP. Kang et al. [49] improved the work in [66] by combining the periodicity and magnitude of P-frame prediction residual for frame deletion and insertion detection. For several reasons, this abnormal periodical increase in prediction residual is difficult to detect. Firstly, prediction residual is content-related and therefore inherent fluctuations in prediction residual images unavoidably conceal the periodic artifacts. Secondly, when the location of frame deletion is close to the end

of a video clip, the period can not be effectively observed. Thirdly, if the content of a tampered video is static, except for a handful of (typically one or two) spikes within a rather small neighborhood of the deletion location, there will be hardly any spike. Hence, Yu et al. [144] fused two features that measure the magnitude of variation in prediction residual and the number of I-MBs instead of checking the periodicity of these artifacts. The work in [1] used quantization effect on I-frame DCT coefficients and P-frame residual errors for frame insertion and deletion detection. It consists of three modules: double compression detection, malicious tampering detection and decision fusion. For double compression detection, an SVM classifier based on first significant digit distribution of I-frame DCT coefficients is used. The quantization effect on P-frame residual errors is used for identifying malicious inter-frame forgery. The decision fusion module classifies input videos into three categories: single compressed, double compressed without tampering and double compressed with tampering.

Wu et al. [138] presented a method for detecting frame duplication and frame deletion tampering using velocity field intensity (VFI). Inter-frame video tampering contributes irregularities in VFI. The inconsistencies in VFIs computed along the vertical and horizontal directions using Particle Image Velocimetry (PIV) is detected by employing generalized extreme studentized deviate (ESD) test. ESD test can detect a maximum of 'n' number of outliers from a univariate data set which follows an approximate normal distribution where the user need to specify the upper bound.

Feng et al. [24] discussed a method for finding the frame deletion point (DP) in a tampered video using the mean motion residuals computed from consecutive video frames. The usage of mean motion residuals helps in reducing the possible false positives in fast motion videos. The statistical properties of a frame at DP will be similar to that of an interference frame. Relocated I-frames (RI), frames with sudden zooming, frames with sudden brightness change and so on are examples of interference frames. RIs are those frames which were encoded as I-frames in 1^{st} compression and later re-encoded as P-frames in 2^{nd} compression after video editing. The fluctuation of distribution of the residual is more in DP than others because of the temporal content difference introduced by tampering. A texture descriptor based method is used for quantifying this fluctuation strength.

The differences in the correlation coefficients of gray values between consecutive frames in a video is presented in [130]. These differences are consistent in pristine videos and abnormal in tampered. It is then normalized and quantized to form a vector for classification using SVM. The work presented in [125] is extended by Gironi et al. [31] using phase change in periodicity for detecting frame addition or deletion.

Local binary pattern (LBP) based method for frame deletion and insertion detection is presented in [148]. In a pristine video, LBPs obtained from consecutive frames will be highly correlated. Inter-frame tampering reduces this correlation at deletion and insertion point. Correlation may also decrease based on the variation of video contents. The Quotients of consecutive correlation coefficients of LBP (QCCoLBP) is used for handling such cases. For detecting the abnormal points, QCCoLBP is processed with Tchebyshev inequality. Multi-Level Subtraction (MLS) based method for detecting frame insertion is presented in [38]. There are 3 levels of subtraction: (1) pixel values in grayscale are subtracted from consecutive frame pairs, (2) consecutive value pairs from first level are subtracted and (3) consecutive value pairs from second level are subtracted.

Inter-frame video tampering detection methods based on Lucas Kanade optical flow computed between consecutive frames are presented in [12, 46, 135]. The optical flow computed from consecutive frames along horizontal and vertical directions are consistent in pristine videos. Inter-frame video tampering may introduce irregularities in OF. A video under investigation is divided into subsequences in [12]. The OF computed between the first and last frames within a sub-sequence is thresholded for identifying the candidate frames of tampering. A binary searching scheme is employed for finding frame insertion point. Frame-to-frame OF in a sub-sequence is computed for frame deletion detection. For each frame, the sum of magnitudes of OFs are computed in [135]. OPVF is estimated for revealing the comparative changes in the OF of a frame corresponding to its immediate neighbours. Gaussian model-based statistical method is employed for anomaly detection due to frame duplication, deletion, and insertion. Kingra et al. [50] utilized prediction residual and OF gradients for detecting frame insertion, duplication, and deletion. Jia et al. [46] presented OF based method for frame

duplication detection. The irregularities in OF is used to find the candidate region of forgery. Then, OF correlation between duplicated frame pairs and video inherent features such as similarity, continuity and regularity are employed for confirming the duplication.

Zheng et al. [151] used Block-wise Brightness Variance Descriptor (BBVD) for frame deletion and insertion detection. The persistence of human vision is utilized here. In pristine videos, the ratio of BBVD between consecutive frames in a specific interval (0.4s) is close to a constant and is consistent; considerable variation indicates tampering.

Yang et al. [142] presented a similarity-analysis method for detecting frame duplication. The frames are decomposed using Singular Value Decomposition (SVD). A video is divided into overlying sub-sequences. The Euclidean distance between the SVD feature of the reference frame which is the first frame in the sequence to that of the rest of the frames in the same sub-sequence is obtained for further processing. Based on the correlation of features, a video containing highly correlated sub-sequences are classified as candidates of duplication. The candidate sub-sequences are verified through random block matching. Zhao et al. [149] used the similarities of Hue-Saturation-Value (HSV) histograms between consecutive frame pairs. The presence of two abnormalities in a similarity curve is regarded as frame insertion or duplication, and one is frame deletion. Based on FLANN matching of the SURF keypoints, frame duplication is distinguished from frame insertion.

In [99], 9 features (four from the ratio of each sub-block, one from the mean of frame/block, and four from the residue of each sub-blocks) are extracted from frames after dividing them into four sub-blocks. Lexicographical sorting groups similar frames. The features of consecutive sorted frames are used for calculating RMSE. Frames having RMSE greater than a threshold are kept as suspicious. If suspicious frames are highly correlated, then they are classified as candidates of duplication. For reducing the false positives, four successive frames following the candidate frames of duplication are inspected for duplication. If these frames are highly correlated, the corresponding video is categorized as frame duplication tampered.

Liu and Huang [67] transformed RGB frames to 2-D opponent chromaticity space for detecting frame duplication, deletion, insertion, and replacement tampering. The correlation of Zernike moments is employed for calculating Zernike Opponent chromaticity moments (ZOCM) from the chromaticity space. The irregular points are identified using the variations in ZOCM computed from consecutive frames. For reducing the false positives, Tamura coarseness feature is used.

Gupta et al. [34] utilized the mean square value of motion energy computed from the difference images at spatial region of interest (SROI) for frame repetition detection. SVM classification with entropy of difference images and average object area are used for frame deletion detection. Frame deletion and insertion detection using the sum of absolute differences between the frames before and after applying the deblocking filter in an H.264/AVC video is presented in [107]. It exhibits better performance on smooth content or fixed Quantization Parameter (QP) video. A method for checking QP is also presented. The relationship between QP and bitrate of a frame is used for finding the QP of the following frame. If the estimated QP differs from that present in the video, the corresponding video is categorized as tampered.

Frame deletion detection using features such as increase in I-MB count, prediction residuals, qs and an estimate of PSNR is discussed in [94]. Shanableh employed SVM, k-NN, and logistic regression over these features for evaluating the feature strength over various machine learning techniques. This method is suitable for AGOP, VBR and CBR coded videos.

2.3.4 Other methods

Video tampering detection methods for upscale-crop forgery are presented in [41, 42] and [97]. Hyun et al. [41] utilized resampling detection method in [29] and Sensor Pattern Noise (SPN) for upscale-crop forgery detection. Its performance is poor in compressed and static surveillance videos. Later, they modified the work using scaling invariance of minimum average correlation energy Mellin radial harmonic (MACE-MRH) correlation filter to reliably unveil traces of upscaling in videos [42]. Two different MACE-MRH correlation filters are designed

from a reference SPN for distinguishing normal and upscale-cropped sequences. The correlation values between SPNs of resampled and normal sequences vary depending on SF. As SF increases, SPN correlation decreases. Singh and Aggarwal [97] utilized pixel correlation and SPN. Even though these methods are successful in detecting upscale-crop forgery, they fail with synthetic zooming (Example 2 in Section 1.2 of chapter 1). Because, the frames are not upscaled with a constant SF. The basic assumption in [42] and [97] is that in a video, a sub-sequence of frames starting from a random frame position is upscale-cropped with a single SF. SPNs computed from natural and synthetic zoomed frames vary from that of normal frames. As SPN is the accumulated noise computed from frames in a sample sequence, motion and lens defocus during camera zoom-in or zoom-out operations will contribute extra noise. Hence, the resulting SPN differs from reference SPN. Also, camera defocus will affect pixel correlations in frames recorded during natural camera zooming and it resembles synthetic zooming. These factors contribute false positives for optical camera zoom in [42, 97].

A method for video tampering detection using features from video file structure is presented in [103]. The basic assumption behind this work is that video editing tools will not allow the users in accessing or modifying its field structures. A database for holding the signatures corresponding to the ordered file structure of video content based on video editing software is created. The video content of the video under investigation is compared to the signatures in the database. This method is successful in tampering detection provided the file structure has a match in the database. It fails when a perpetrator tampers a video using a custom technique (without using any state-of-the-art tools).

2.4 Image Resampling Detection

The lack of works in video resampling detection has forced us to go through some works in image resampling detection [29, 51, 52, 53, 70, 71, 80]. Popescu and Farid [80] identified that the interpolation operations in image resampling will introduce linear dependencies between adjacent pixels in resampled image. Interpolation is the process of finding pixel values at intermediate pixel locations which are not present in its original version. They estimated the linear correlation

between each pixel with its neighboring pixels in the resampled image and their probability of correlation using EM algorithm. The p-map (correlation probability map) will exhibit periodic peaks in its frequency domain if the image under investigation is resampled. The strength of linear dependence varies periodically, which itself depends on the resampling parameters. This method is complex and time consuming due to EM algorithm. Also, it is sensitive to noise and compression artifacts. Kirchner modified [80] by replacing the EM algorithm with linear filter in [51], linear row and column predictors in [52] to reduce its computation. Kirchner and Gloe [53] later extended the work in [51] for resampling detection in recompressed images. Gallagher [29] identified that the variance of second order derivatives of interpolated images exhibit periodic patterns. It is then generalized by the authors of [70, 71] that the variance of n^{th} order derivatives of interpolated images exhibit periodic patterns. The periodicity is investigated using DFT and Radon Transformation in [29] and [70, 71] respectively.

Noise and blocking artifacts generated by lossy compressions like JPEG affect resampling detection. High compression ratio may introduce more noisy pixels. Block-based lossy compression techniques introduces sharp transitions at block boundaries between neighboring blocks. This blocking artifacts will introduce periodic peaks which when coincides with that of resampling leads to the failure of algorithms. This paves way for anti-forensic approaches on the methods in [29, 51, 52, 53, 70, 71, 72, 80, 81]. Using a high compression rate will help in hiding the periodic patterns of actual resampling on that from blocking artifacts. Earlier methods ignored all such peaks at frequencies $(k/8, l/8), (k, l) \in \{-4, ..., 4\}^2$ to avoid JPEG peaks due to blocking artifacts. Hence, when these peaks overlap with tamper artifacts, the candidate image will be misclassified as pristine. The noise introduced by compression will also function as anti-forensics, as it decreases the detection performance of these algorithms. The method in [29] fails when the original image is resampled with an interpolation factor of 2 where the phase of variance signal is preserved. Also, it fails when the resampled region is rotated or skewed.

2.5 Anti-forensic Techniques

Anti-forensic or counter forensic techniques are those which are designed to mislead forensic analysis by erasing or falsifying fingerprints (tamper traces or artifacts) left by editing operations. Anti-forensic tools and techniques are also used as privacy protection measures. Just as the fact that digital editing processes leave traces, anti-forensic techniques also leave its traces. Knowing this, researchers have come up with methods for countering anti-forensics. Those techniques designed to reveal the traces of anti-forensic activities are called counter anti-forensic techniques. A perpetrator must balance the trade-off between complete removal of forgery traces and introduction of new evidence from anti-forensic manipulations for attaining imperceptible forgery. On the other hand, investigators must maintain a trade-off between the accuracies of forgery detection and anti-forensics.

Stamm and Liu [105] discussed an anti-forensic approach for frame deletion and insertion detection presented in [131]. When frames are shifted from one GOP to another, it introduces periodic spikes in the DFT of P-frame prediction error [131]. The anti-forensic operation in [105] modify MPEG encoding process for converting the P-frame prediction error sequence to a target sequence. This will remove the temporal fingerprint from P-frame prediction error sequence. The error value of a given P-frame is increased to the target by changing the frame's predicted value in a manner that increases the prediction error. MVs of selected MBs of those P-frames are set to 0 for obtaining new prediction errors. Stamm and Liu [104] proposed a counter anti-forensic technique which will detect the unusual MVs with zeros by comparing the MVs of video under investigation with its estimated true MVs. Kang et al. [49] also proposed a counter method for detecting the anti-forensics discussed in [105] by comparing the true prediction error computed from the video under investigation to that stored in it.

Su et al. [107] discusses an anti-forensic method where the MB types of frames and quantization indices before and after targeted frames are recorded as reference after decoding a video, to limit the coding modes of the targeted frames. In second round encoding, the indices are adjusted along with the MB types and residuals so that the edited video seems to be a normal one.

Frame duplication is typically detected using the repetition of a statistical property [65, 99, 135]. When the original frames used to accomplish frame duplication are reordered before placing at another temporal location of the same video, these methods fail. Hence, frame shuffling forms a suitable anti-forensic approach on frame duplication detection as researchers usually exploit features extracted from the repetition of frame sub-sequence in the exact same order.

Up-scale crop tampering is a suitable anti-forensic approach on frame insertion, deletion, duplication or shuffling. It helps counterfeiters to perform video tampering without performing any of the traditional inter-frame or intra-frame forgeries. Up-scale crop forgery enable them in removing any unwanted criminal activities happened on the border regions of surveillance FOV. In such videos, outer frame regions are cropped so that activities at the border of frames are removed. These cropped frames are up-scaled so that they can have the same resolution as that of original untouched frames in video. With small up-scale factors, these forgeries may not be detected with inter-frame forgery detection methods. If they detect, then it will have 2 tamper locations, one at the beginning and other at the end of up-scale cropped frames. Inter-frame forgery detection methods will confuse it for frame insertion. Assuming that the frames at suspicious locations are dissimilar in frame insertion, the methods in [12, 135] use thresholding approach for insertion detection. As the candidate frames of forgery are taken from the same video, this is violated.

A simple anti-forensic approach on upscale-crop forgery detectors in [41, 42, 97] is synthetic zooming. That is, instead of performing upscale-crop with a fixed SF on all frames, SF can be incremented gradually. Then, the resulting frames resemble to those recorded during camera zooming operation. For example, consider an event occurring near the border of FOV need to be deleted. The perpetrator can perform upscale-crop forgery in those frames. If he is in a situation that he has to use a large scale factor (> 1.3, quarter portion of the frame will be removed after upscale-cropping) for removing the event, then it may be noticeable while watching the video. Rapid scale change at immediate neighboring frames (frame that precedes and follows upscaled frames) with large scale factor will introduce visible artifacts. To avoid this, the frames preceding and following the event can be upscaled with variable (gradually decreasing) scale factor for

smooth upscaling. If the frame sequence corresponding to an event need to be deleted completely, then perpetrators can also perform frame duplication with synthetic zooming. That is, a sequence of frames from former or later instance of time can be copied and pasted for filling the gap of deletion. These copied frames can be processed for synthetic zooming before or after pasting it to the new location. It works as an anti-forensic technique on frame duplication detection as the synthetic zoomed frames will no longer share any statistical properties with their original frame sequence. With the availability of video editing tools such as Adobe After Effects, one can easily perform such synthetic zooming.

The limitations discussed in section 2.9 can be utilized by perpetrators for developing anti-forensic techniques.

2.6 Camera Tampering Detection

We explore various camera tampering detection methods in the literature based on the movement of the surveillance camera in an intended FOV: (1) Static in FOV, and (2) Panning in FOV.

2.6.1 In Static Surveillance Systems

Normally, a background model regarding the intended FOV is maintained for determining the camera tamper events present in new incoming frames. Some methods keep a single frame as reference. But, it may contribute to false alarms subjected to variations in the intended FOV such as illumination changes. To overcome this by creating a better background model, a group of frames or information collected from a frame sub-sequence may be used. Zero-mean normalized cross correlation, edge information, and decrease in entropy computed between the incoming frames and background model are employed for tamper detection in [30]. The reduction in high frequency components (edges) and comparison of histograms between the background model and incoming frames are taken as features of detecting camera tampering activities in [4, 21, 55, 64, 91, 95]. Aksay et al.[4] used histogram peak comparison for detecting camera occlusion and reduction of high frequency wavelet coefficients for camera defocus detection. The

histogram of occluded frames are skewed towards the black end of grayscale. The small scale details of FOV may not be recorded during camera defocus which in turn decreases the high frequency components in incoming frames. The detection of camera occlusion, displacement and defocus anomalies using histogram comparison and high frequency components from FFT are presented in [91]. Kryjak et al. [55] extended the work in [91] with FPGA implementation. Edge components, block matching, and entropy are used in [21]. Histogram comparison and edge difference based tamper detection technique implemented on a TI DAVINCI DM6437 Digital Signal Processor is presented in [64]. The works presented in [4, 21, 55, 64, 87, 91] fail to detect camera tamper when occlusion is performed using textured objects and generate false alarms when parts of the intended FOV is occluded because of crowd or large objects passing through the FOV. Shih et al. [95] presented a two stage process for camera tampering detection. The first stage is for fast processing of incoming frames. Sample points computed from edge gradients of background are compared to that in incoming frames for tamper detection. The second stage is for reducing the misclassification occurring in first stage. It uses features from the scene structure of the whole frame. Ribnick et al. [87] kept incoming frames in newer and older pools based on the order of arrival. The dissimilarities between the frames in these pools are calculated for tamper detection. The standard deviation and edge energy computed from frame patches and frame as a whole with Kalman filtering is presented in [136]. An adaptive background codebook model based method is presented in [123]. The SIFT key points computed from background model and incoming frames are compared based on its position change and reduction in number for tampering detection in [143]. Huang et al. [39] discussed methods for detecting camera defocus, occlusion, screen shaking, motion, screen flickering, fogging, and color cast based on high-frequency information, histogram comparison, edge details, and brightness analysis.

2.6.2 In Panning Surveillance Systems

Camera tampering detection using Speeded Up Robust Features (SURF) key points is presented in [122]. It is applicable on both static and panning surveil-

lance systems. Blockwise HOG comparison between background and incoming frames, key point reduction, frames captured per panning cycle, and key point displacement are the features used. This method suffers from false alarms.

2.7 Benchmark Datasets

In this section, we discuss video and camera tampering datasets that are publicly available. Subsection 2.7.1 deals with video tampering detection datasets. Subsection 2.7.2 deals with video datasets containing pristine videos that we used for creating tampered videos to test our proposed methods. Camera tampering detection datasets on static and panning surveillance systems are discussed in Subsections 2.7.3 and 2.7.4 respectively.

2.7.1 Video Tampering Datasets

A freely available dataset for video forensics called Surrey University Library for Forensic Analysis (SULFA) [83] is the first of its kind that focuses on source camera identification and integrity verification. It contains 150 original videos captured using 3 different cameras with a resolution of 320×240 and frame rate of 30 fps. Video dataset provided by Bestagini et al. [9] contains 20 videos for copy-move forgery detection. It has 10 pristine and 10 forged videos. A few videos are from SULFA. Also, the resolution and fps of videos in this dataset are same as that in SULFA.

Ardizzone and Mazzola [5] developed a tool for creating tampered videos. It gives the user an option to select the object or region to be copied. It also requires the number of frames to be specified so that its motion can be tracked. Further, the users should specify the destination where the copied object is to be pasted. The transformation parameters can also be specified if copy-move forgery with affine transformation has to be performed. The copy-paste can be carried out in the same or different video. Tracking of the selected object is done using SURF keypoint descriptors. The trajectory of objects can also be modified. Blending operation is employed to make tampering appear more realistic.

Due to shortage of datasets, most researchers make modifications to downloaded or recorded videos to suit their research requirements for training and/or testing.

2.7.2 Pristine Video Datasets downloaded for creating Tampered Videos

Public datasets devoted to inter-frame video tampering detection and synthetic zooming detection are not available. We have created our own datasets for these from pristine videos by downloading those from [118, 145], PETS2001 [79], PETS2007 [26] and PETS2009 [25]. Bridge-Close, Mobile, Akiyo, Highway, Waterfall, Coastguard, Silent, Carphone, Galleon, Mother and Daughter, Claire, Tempete, Container, Sign-Irene, News, Foreman, Pamphlet, Grandma, Hall, Husky, Paris, Bowing, and Bridge-Far are the videos downloaded from [118, 145]. These videos are in YUV uncompressed format with CIF (352×288) resolution. The resolution of videos from PETS2007, PETS2009 and PETS2001 are 720×576, 768×576, and 768×576 respectively. Table 2.5 provides the summary of these datasets.

2.7.3 Camera Tampering Dataset on Static Surveillance system

Dataset provided by Saglam and Temizel [91] is downloaded for testing the proposed method for camera tampering detection in static surveillance systems. This dataset consists of a range of images to reflect real-life camera tampering scenarios. We received only 39 videos out of the original 54 sequences used in [91]. The frame rate and resolution of the videos in this dataset are 25 fps and 320×240 respectively. There are 13 videos with camera defocus anomaly, 12 with camera displacement, and 17 with camera occlusion anomaly. Some of the videos in this dataset are having multiple camera tamper events. The summary of the number of videos in each camera tampering category is given in Table 2.6.

Table 2.5: Summary of videos taken from pristine video datasets

Datasets	Number of Videos taken	Resolution
Trace	23	(352×288)
PETS 2001	20	768×576
PETS 2007	16	720×576
PETS 2009	104	768×576

Table 2.6: Summary of the static camera tampering dataset from [91]

Video Type	Number of Videos Mentioned	Number of Videos Received
Defocus	35	13
Occlusion	40	17
Displacement	12	12

2.7.4 Camera Tampering Dataset on Panning Surveillance system

The public dataset from [122] for active or panning surveillance camera is made available by the VAP's (Visual Analysis of People) laboratory. The proposed method for camera tampering detection on panning surveillance system is tested using this dataset. It is the single available dataset in this category. The frame rate and resolution of the videos in this dataset are 25 fps and 720×576 respectively. It consists of 80 videos of which 4 are non-tampered videos. The remaining 76 videos have single tamper event which may be due to camera occlusion, camera defocus or camera displacement. The videos corresponding to each camera tamper type are available as separate files. The surveillance camera models used for video recording are Sony EVI-G21 and Sony EVI-D31. The videos are recorded from three different locations: a park area, a backyard, and

an indoor foyer. The settings of the surveillance cameras are kept in auto mode for creating this dataset. The summary of the number of videos in each camera tampering category is given in Table 2.7.

Table 2.7: Summary of the panning camera tampering dataset from [122]

Video Type	Number of Videos
No tamper	4
Defocus	31
Occlusion	14
Displacement	31

2.8 Performance Measures used

The performance measures used for evaluating the detection performances of proposed methods are recall (R), precision (P), accuracy (A), and $F_1 score$ as follows:

Recall: It is the fraction of positive instances identified among actual positive instances. It is also known as Sensitivity or True positive rate (TPR).

$$R = \frac{TP}{TP + FN} \tag{2.1}$$

Precision: It is the fraction of correct positive instances among the identified positive instances.

$$P = \frac{TP}{TP + FP} \tag{2.2}$$

Accuracy: It is a metric for evaluating classification models. It is the fraction of true predictions among the total number of cases examined.

$$A = \frac{TP + TN}{TP + FN + TN + FP} \tag{2.3}$$

$F_1 score$**:** It is the harmonic mean of precision and recall. Precision is used together with recall to provide a single measurement of performance of the

system for the positive class.

$$F_1 score = \frac{2 \cdot P \cdot R}{P + R} \tag{2.4}$$

where '$\cdot$' denotes multiplication.

where TN, FP, TP and FN are the number of true negatives, false positives, true positives and false negatives respectively of a binary classification test. TP is a result which indicates that a given condition exists when it does. TN indicates a result that a condition does not exists while in fact it does not. FP is a result which indicates a given condition exists when it does not. FN indicates a result that a condition does not exists while in fact it actually exists. Precision and recall are evaluated for understanding the measure of relevant classifications.

2.9 Limitations of Existing methods

Video tampering detection

- Frame duplication detection methods fail in static scene videos when duplicated frames are shuffled before pasting them in another temporal location. Researchers used to leverage the repetition of frame sequence or its properties in the exact same order for frame duplication detection. Frame shuffling changes this order of frames and thereby affecting the detection system.
- Most of the methods fail in AGOP structure videos. The literature which utilizes tamper footprints from the compression domain alone may make us rely on the periodicity of artifacts (restricting to fixed GOP videos). Modern video codecs support AGOP structure where the number of frames in a GOP varies depending on the content of video to maintain its quality.
- Sudden zooming of lens have a negative impact on the performance of most algorithms discussed in this chapter. Such videos may get misclassified as tampered.

- Synthetic zooming can be applied as an anti-forensic technique on inter-frame video tampering detection. To a viewer, these videos appear as genuine as it is recorded as part of natural camera zoom. We can perform inter-frame video tampering such as frame duplication and frame shuffling with them. Hence, it is ideal to classify videos with zooming as pristine or tampered after analyzing whether their zoomed frames are created synthetically or naturally.

- To hide an object in video, perpetrators can use other objects present in the same video. These objects can be subjected to affine transformations before pasting. There is not much in work in the literature of video tampering detection discussing this issue.

Camera tampering detection

- The reliability of a camera tamper detection system depends on low false alarm rate. Reliability is a subjective aspect based on the operator's psychology as he/she may overlook future alarms assuming wrong ones. Countering false alarms is a challenge which camera tampering detection algorithms fail to overcome in the literature.

- The current scenario is that the general public is interested in installing PTZ cameras rather than static cameras. We found only one article in the literature on camera tampering detection in PTZ surveillance systems.

In this thesis, we addressed the first four limitations of video tampering detection techniques discussed above. We have also developed techniques for reducing the drawbacks of camera tampering detection.

Chapter 3

Inter-Frame Video Tampering Detection

In this chapter, we describe the proposed methods to detect inter-frame video tampering such as frame deletion, frame insertion, frame duplication and frame shuffling using artifacts of video tampering from the spatio-temporal and compression domains. If spatio-temporal artifacts alone is taken for tampering detection, then it may increase false positives. Similarly, if we take compression domain artifacts, then it may restrict the application of those methods on fixed GOP videos as most of them rely on the periodic occurrence of artifacts. Therefore, we took artifacts of video tampering from spatio-temporal and compression domains. Inconsistencies in velocity fields computed from consecutive frames are used as spatio-temporal artifacts, while, variations in prediction regarding the MB types of P-frame is used as a compression domain artifact. Generalized extreme studentized deviate (ESD) algorithm [44] is used to find the abnormalities in velocity field sequence that occur as a result of inter-frame video tampering. The candidate locations of video tampering identified using ESD are inspected for compression artifacts. If it exists, videos are classified as tampered. Furthermore, we propose methods to identify the kind of inter-frame video tampering employed at these tamper locations. The main contribution of our work is that we have developed a method for frame shuffling detection and identifying the original frames involved in it which is an unresolved issue. Also, a new method for zooming detection is proposed to decrease the false positives on sudden zooming.

The rest of this chapter is organized as follows. Section 3.1 provides the background concepts for understanding this work. Section 3.2 describes the proposed methods for inter-frame video tampering detection. Section 3.3 deals with experimental results and discussion. Section 3.4 provides conclusions and future research directions.

3.1 Preliminaries

This section deals with the discussion on two artifacts adopted in this work.

3.1.1 Variation of Prediction Footprint

A video may be recompressed for storage or transmission purposes or after video editing. I-frames in the first compression may get re-encoded as P or B frames. It can occur in two ways: (1) GOP sizes used in 1^{st} and 2^{nd} compression differ, and (2) video tampering. Such I-frames are termed as Relocated I-frames (RI). Let us consider a scenario where a video is captured and it underwent first compression in capturing device with a fixed GOP size (G_1). This compressed video is decompressed to raw uncompressed format and applied another compression with different GOP size G_2 such that $G_1 \neq G_2$. This introduces variation in the number of MB types of P and B frames previously encoded as I-frames in 1^{st} compression. Suppose $G_1 = 9$ and $G_2 = 12$, then 10^{th} frame which is an I-frame in the first compression will get converted to P or B frame depending on the combinations of P and B frames used in the GOP. If GOP structure of first and second compressions are "IBBPBBPBB" and "IBBPBBPBBPBB" respectively, then 11^{th} frame in second compression is a P-frame. This P-frame is an I-frame in first compression. The macro-block types in a P-frame are of 3 types: P-MB, S-MB and I-MB. The second encoder cannot code most of the contents of an RI with S-MB or P-MB. I-frame comprises more detailed information. If it is encoded with S-MBs and P-MBs, then proper reconstruction of RI from its reference frames is not possible. Hence, the second encoder codes P-MBs and S-MBs in homogeneous regions as I-MBs; and S-MBs in static regions as P-MBs. This process affects the distribution of macro-block types in RI frames compared

to its neighboring frames. It increases the number of I-MBs and decreases the number of S-MBs in RI frames. This decrease of S-MBs and increase of I-MBs in an RI frame opposed to its neighboring P-frames is termed as Variation of Prediction Footprint (VPF). Vazquez-Padin et al. [125] and Gironi et al. [31] explained this concept. The strength of VPF, $v(n)$ is calculated as:

$$v(n) = |(i(n)-i(n-1))\cdot(s(n)-s(n-1))|+|(i(n+1)-i(n))\cdot(s(n+1)-s(n))| \quad (3.1)$$

where $s(n)$ and $i(n)$ denote the count of S-MBs and I-MBs respectively present in the current RI (n^{th}) frame; $s(n-1)$ and $i(n-1)$ denote the count of S-MBs and I-MBs respectively present in the P frame previous to n; and $s(n+1)$ and $i(n+1)$ denote the count of S-MBs and I-MBs respectively present in the P-frame following n. The strength of $v(n)$ is high for RI frames. This artifact appears periodically in fixed GOP videos. In such videos, the relocation of an I-frame causes the relocation of all its following I-frames. When a sub-sequence of frames in a fixed GOP video undergo inter-frame tampering and is re-encoded where the number of frames in the sub-sequence is not an integral multiple of GOP size, the frames of the tampered video will be re-organized in new GOPs in second compression after editing. This leads to relocation of I-frames. Gironi et al. [31] utilized VPF for frame insertion and frame deletion detection. When an entire GOP or its integral multiples are tampered and the GOP size of 2^{nd} encoding is the same as that used in 1^{st} encoding, I-frame of first compression remain as I-frames in second compression also. No I-frames will be converted to RI in this case and hence, the method fails. Another limitation is that it works in videos having fixed GOP structure only. The count of frames in consecutive GOPs of an AGOP video is not constant. Hence, VPF artifacts will not occur periodically in AGOP videos.

3.1.2 Velocity Field Intensity

Particle Image Velocimetry (PIV) [33] technique is employed for computing the velocity field. PIV is a non-intrusive analysis technique for qualitative and quantitative flow visualization. It calculates particle displacement of groups of particles. For this, the correlation of small sub-images also called as interrogation areas at

time t and $t+1$ are computed. Wu et al. [138] measured velocity field using cross-correlation of FFT window deformation on one-pass 16×16 interrogation areas with 75% overlap factor:

$$R_c(u,v) = \mathcal{F}^{-1}(\mathcal{F}(I(x,y,t))[\mathcal{F}(I(x,y,t+1))]^*) \tag{3.2}$$

$$(u(x,y,t), v(x,y,t)) = argmax_{u,v} Re\{R_c(u,v)\} \tag{3.3}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the $2-D$ FFT and inverse FFT operators respectively; $I(x,y,t)$ and $I(x,y,t+1)$ denote interrogation windows corresponding to $(x,y)^{th}$ location in frames at time t^{th} and $(t+1)^{th}$ respectively; $*$ denotes complex conjugate function. $(u(x,y,t), v(x,y,t))$ denotes the displacement vector (or velocity vector) between the two interrogation windows under consideration. Re gives the real part of its parameter. Velocity field intensity (VFI) of a frame at time t is expressed as:

$$VFI_h(t) = \sum_x \sum_y |u(x,y,t)| \tag{3.4}$$

$$VFI_v(t) = \sum_x \sum_y |v(x,y,t)| \tag{3.5}$$

where $VFI_v(t)$ and $VFI_h(t)$ denote VFIs computed along vertical and horizontal directions respectively.

To identify tampered videos, Wu et al. [138] used inconsistencies in velocity. Due to inter-frame tampering, non-adjacent frames (temporally distant) become neighbors at tamper locations. Let us discuss this with frame deletion. Frames from a temporal location g, $g+1$, ..., $g+g'$ are to be deleted. After deleting these frames, in the tampered video frames at locations $g-1$ and $g+g'+1$ become new neighbors. In case of a frame duplicated video, the first and last frames of the frame duplication sub-sequence are temporally distant from its new neighboring frames. The sub-sequence used to perform frame duplication forgery is recorded at a different instance of time. Hence, the frame that immediately precedes 1^{st} frame of the duplicated sub-sequence in a tampered video is temporally distant in its original non-tampered video. Likewise, the frame that immediately follows the last frame of a duplicated sub-sequence in the tampered video is temporally distant in its original non-tampered video. When non-adjacent frames become neighbors in a tampered video, it contributes to abnormalities in VFI. Camera

noise may sometimes reduce the VFI of a frame. The max-sample technique is employed for reducing the false positives that may occur in such cases. It is the highest VFI value on a window of 3 frames.

$$SVFI_h(t') = max(VFI_h(t-1), VFI_h(t), VFI_h(t+1)) \tag{3.6}$$

$$SVFI_v(t') = max(VFI_v(t-1), VFI_v(t), VFI_v(t+1)) \tag{3.7}$$

where $SVFI_h(t')$ and $SVFI_v(t')$ denote horizontal and vertical max-sampled VFI sequences respectively. To reveal abnormal peaks, Wu et al. [138] computed relative factor sequences from SVFIs as:

$$RF_h(t') = \frac{SVFI_h(t'+1) + SVFI_h(t'-1)}{SVFI_h(t'+1) \cdot SVFI_h(t'-1)} \cdot SVFI_h(t') \tag{3.8}$$

$$RF_v(t') = \frac{SVFI_v(t'+1) + SVFI_v(t'-1)}{SVFI_v(t'+1) \cdot SVFI_v(t'-1)} \cdot SVFI_v(t') \tag{3.9}$$

where $RF_h(t')$ and $RF_v(t')$ denote the horizontal and vertical relative factor sequences respectively. To find the abnormal peaks in $RF_h(t')$ and $RF_v(t')$, Wu et al. [138] employed ESD algorithm. The feature used for identifying inter-frame tamper type is the count of abnormal peaks in these two sequences. Based on this, an input video can be classified as: (i) pristine if no peaks, (ii) frame deletion if there exists a single abnormal peak in either $RF_h(t')$ or $RF_v(t')$, and (iii) frame duplication if there exist two abnormal peaks in either $RF_h(t')$ or $RF_v(t')$. However, there are cases where videos with frame duplication forgery may not generate two peaks in its RF sequences. Case 2 in Section 3.2.3 discusses this scenario. Also, the count of abnormal peaks in either $RF_h(t')$ or $RF_v(t')$ as the sole feature for inter-frame video tamper type classification may contribute false positives.

3.2 Proposed methods for Video Tamper detection

The flowchart of the proposed method for inter-frame video tampering detection is presented in Fig. 3.1. An input video subjected to inter-frame tampering detection undergoes three different processes which are represented as three different

blocks (brown, green and blue color dashed lines). The brown block determines temporal candidate locations (or frame positions called tamper points, t_p) of inter-frame tampering using inconsistencies in VFI obtained from consecutive frames in an input video. The method for computing t_p is displayed in the flowchart followed by an explanation in subsection 3.2.1. The green block validates the recognized tamper locations. $Cardinality(t_p)$ indicates the number of tamper points. If velocity field is consistent, then abnormal points are absent in SVFIs. Hence, $Cardinality(t_p) = 0$ indicates that the video under investigation is pristine. If $Cardinality(t_p) \neq 0$, it indicates that the video under investigation may be tampered. Based on $Cardinality(t_p)$, the following three cases are considered:

- Case 1: $Cardinality(t_p) = 2$
  - indicates video is tampered
  - tamper type can be frame duplication, frame insertion or frame shuffling
- Case 2: $Cardinality(t_p) = 1$
  - indicates video is tampered
  - tamper type can be frame duplication, frame deletion or frame shuffling
- Case 3: $Cardinality(t_p) = 0$
  - video under investigation is genuine

We examine the presence of variations in the distribution of MB types in the 1^{st} P-frame at or after t_p for validating the tamper points as described in Subsection 3.2.2. As we focus on abnormalities, sudden zooming using camera lens adjustments may insert abnormalities in VFI and MB prediction types. The first and last frames recorded during camera zooming activity may contain sudden changes when compared to those that immediately precedes and follows these frames respectively. Therefore, videos comprising zoomed frames may get wrongly classified as frame insertion. To reduce such false positives, we present a zooming detection method which is discussed in Subsection 3.2.4. The frames that immediately precedes and follows t_p are taken for zooming detection. If they

are candidates of zoom-out or zoom-in operations, then we apply techniques for inter-frame video tampering type classification. Frames recorded as part of sudden camera zooming activity will not appear on both sides of t_p. It occurs only on one of its sides. It denotes the beginning or end of a camera zooming activity at t_p. If it occurs on both sides, then they are not candidates of sudden zooming. The blue color block outlines the operations performed for inter-frame video tampering type classification. We took $Cardinality(t_p)$ as the feature for conducting various tests (Subsection 3.2.3) on the frames at or in between tamper positions for tamper type classification.

3.2.1 Estimating the Velocity Field

The velocity field is computed by employing PIV method [120, 121]. Velocity field estimation using cross-correlation with FFT window deformation have three-passes (64, 32 and 16 square pixel interrogation windows for 1^{st}, 2^{nd} and 3^{rd} passes respectively) with 75% overlap factor in 1^{st} and 50% in 2^{nd} and 3^{rd} passes. One-pass interrogation induces loss of information in particle displacement. As a result, in the correlation matrix, there will be an increase in background noise. Further, it complicates velocity peak detection [120] affecting the accuracy of the proposed inter-frame tamper detection method. Velocity estimation by optimizing the displacement in multiple passes (iterative manner) [93, 137] can overcome this drawback. Interrogation windows in next pass are translated based on the predicted displacement from the previous pass. The prediction error is minimized iteratively. In the initial iteration, the predictor is initialized to 0 as the displacement information is not available. The window size is halved in subsequent passes. The displacement information from the last pass is used for predicting the position and translation of interrogation windows in $(t+1)^{th}$ frame. After cross-correlation, the difference between predicted and obtained displacement is used for computing the prediction error. This is used for refining the prediction for subsequent pass.

Horizontal and vertical VFIs ($VFI_h(t)$ and $VFI_v(t)$) are calculated from Eq. (3.4) and Eq. (3.5). If a video contains L frames, then the length of VFI sequence is $L-1$ where $t \in [1, L-1]$.

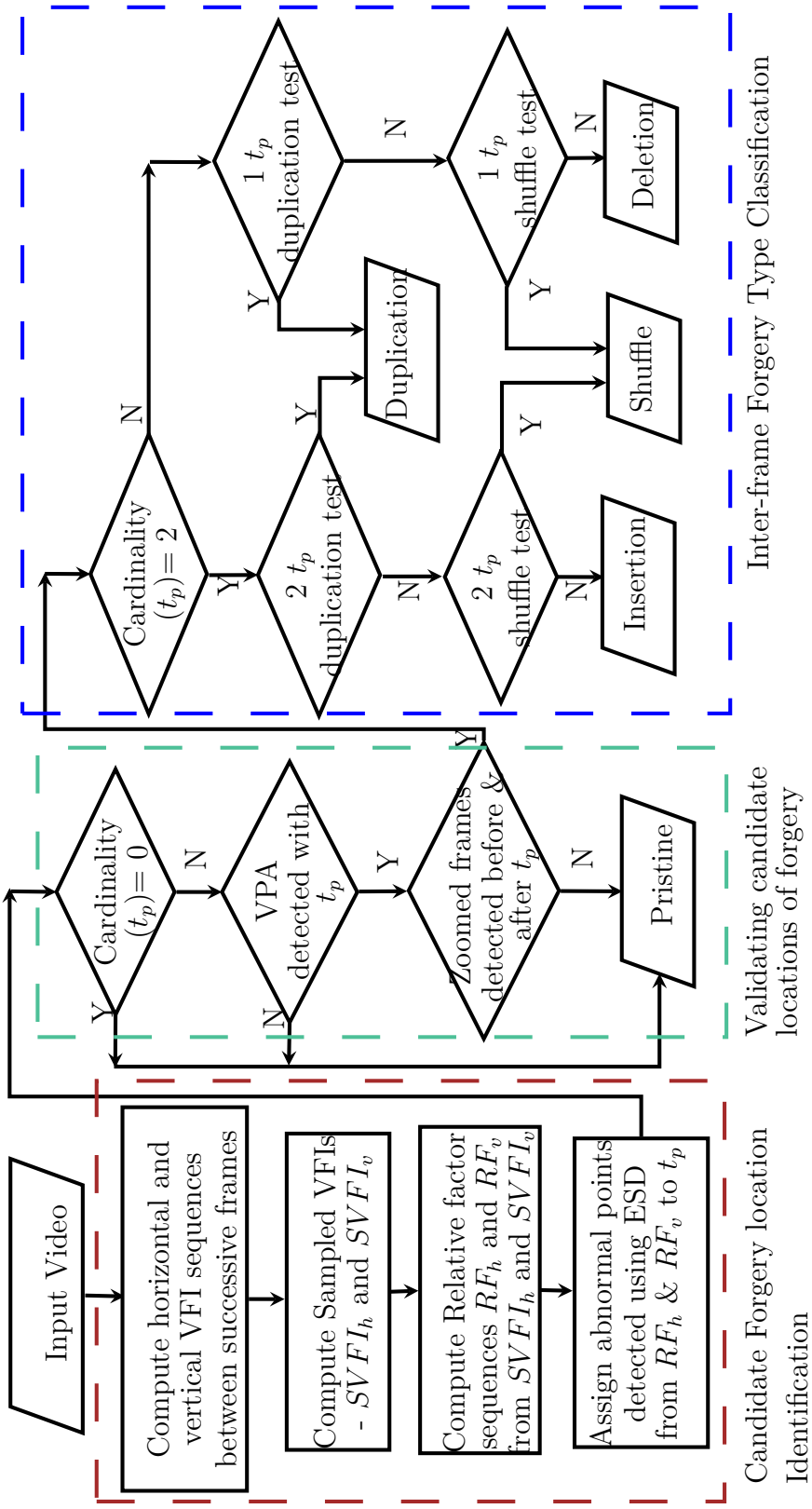


Figure 3.1: Flowchart of the proposed method to detect inter-frame video tampering.

To remove low VFI frames occurring due to camera coding errors, the max-sample method is used. It causes high similarity among adjacent frames. VFI sequences are split into groups of 3 frames and the maximum value in each group is used as the sampled VFI. The max-sampled sequences, $SVFI_h(t')$ and $SVFI_v(t')$ are estimated from Eq. (3.6) and Eq. (3.7) respectively. The length of SVFI sequences are $\lfloor (L-1)/3 \rceil$. Despite the fact that tampered videos are visually impeccable, we can observe abnormalities in VFI sequences. For highlighting these abnormalities in SVFIs, relative factor sequences are computed using Eq. (3.8) and Eq. (3.9) in respective directions. The abnormal peaks (called abnormal points) in RF sequences are estimated using ESD algorithm [44]. As SVFIs are used for computing RF sequences, the abnormal points obtained are not the frame numbers of inter-frame video tampering. We represent the frame numbers of a tampered video where artifacts exist as tamper points, t_p:

$$t_p(k) = (p(k) + 1) \cdot 3 + mod(L - 1, 3) \tag{3.10}$$

where p denotes the set of abnormal points found using the ESD algorithm, $p(k)$ is the k^{th} abnormal point in p, and $t_p(k)$ denote the frame number obtained from $p(k)$.

3.2.2 Variation of Prediction Artifact

The tamper points identified from RF sequences of velocity field are observed for variations in the number of MB types to reduce false positives. As VFI is a spatio-temporal feature, it is more sensitive to noise and lighting conditions than compression domain features for video tampering detection. Hence, validating the frames which cause spatio-temporal artifacts with compression domain artifact will reduce the misclassification of pristine videos as tampered thereby reducing the false positives.

The 1^{st} P-frame at or after t_p may not have the same reference frame that it had on initial compression for its next compression after inter-frame tampering. This may because of: 1) non-existence of the particular reference frame because of frame deletion tampering or 2) time gap between these frames induced by frame insertion or duplication. This disturb MB distributions in these P-frames

by increasing I-MBs and decreasing S-MBs. This is called Variation of Prediction Artifact (VPA):

$$[i(n-1) < i(n)] \wedge [i(n) > i(n+1)] \wedge [s(n-1) > s(n)] \wedge [s(n) < s(n+1)] \quad (3.11)$$

where $s()$ and $i()$ denote the count of S-MBs and I-MBs respectively in the first P-frame at or after t_p denoted by n. $(n-1)$ and $(n+1)$ indicate the P-frames that immediately precedes and follows n. If t_p itself is a P-frame, then n is same as t_p. Otherwise, the 1^{st} P-frame that comes after t_p is used for evaluating Eq. (3.11). VPA that stems from the unavailability of the original reference frame is employed here.

Unlike the methods in [125] and [31], the periodicity of VPA is not required. Therefore, our approach works on fixed as well as adaptive GOP structure videos. Another scenario to be considered is that the frame at t_p is an I-frame. We cannot employ Eq. (3.11) in this scenario. Video codecs like H.264 allot new GOP when it cannot relate a frame to an existing GOP because of its content variations. This occurs as a result of frame tampering in here. Hence, we handle such videos as tampered.

3.2.3 Inter-frame Video Tampering Type Detection

A pristine video does not contain inconsistencies/abnormalities in its VFI sequences. Figure 3.2 shows the horizontal and vertical VFI, SVFI and RF sequences of the "Container" pristine video from [145]. Figure 3.2 (a), (c), and (e) are the horizontal VFI, SVFI and RF sequences. Similarly, Fig. 3.2 (b), (d) and (f) are the vertical VFI, SVFI and RF sequences. Figure 3.3 shows the horizontal and vertical VFI, SVFI and RF sequences of the "Bridge-Close" pristine video from [145]. Figure 3.3 (a), (c), and (e) are the horizontal VFI, SVFI and RF sequences. Similarly, Fig. 3.3 (b), (d) and (f) are the vertical VFI, SVFI and RF sequences. These patterns look almost uniform. There are no abnormal peaks.

A video having abnormal peaks in its VFI sequences is classified as a tampered video if there exist VPA in its t_p. In this work, we took at most 2 tamper points leading to the following cases:

Case 1 RF sequences contain 2 abnormal peaks (t_p). VPA also exists in these t_p. Such videos are classified as tampered. It may be because of frame duplication, shuffling or insertion. These tampering convert 2 distant frames in a pristine video to neighboring frames in tampered video. It occurs in 2 positions, one at the beginning and other at the end of duplicated or inserted frames. Figure 3.4 show VFI and RF distributions of Container video from [145] with 2 t_p. Frames 51 to 75 are duplicated from 76 to 100 after deleting the original frames. 76^{th} frame in forged video is 51^{th} frame in pristine video which is 24 frames away from 75^{th} frame. Similarly, 100^{th} and 101^{th} frames in tampered video are actually 24 frames away in pristine video. The time gap between these new neighboring frames at t_p affect the distribution of VFIs. VFI and RF sequences obtained on this tampered video is shown in Fig. 3.4 (e), (f), (g) and (h). On comparing these sequences with those in Fig. 3.2 (a), (b), (e) and (f) of its pristine video, it can be observed that two abnormal peaks are present in VFI and RF sequences.

2 t_p Duplication Detection The tamper points (t_{p1} and t_{p2}) may be the beginning and ending of frame duplication. The frames in between these tamper points are considered as candidate clip, C which is been copied from somewhere in the same video and pasted at this particular position. For finding its original frames which are replicated, we compute the correlation of VFI sub-sequences with the VFI sequence of C in a sliding window fashion overlapping with $(l-1)$ frames where l is the number of frames in C. The number of frames in each sub-sequence is equal to l. Figure 3.5 is a pictorial representation of this. The frame sub-sequences before t_{p1} and after t_{p2} are taken for correlating with C. The frames enclosed in boxes (dashed, dotted, and dash-dot lined boxes) in Fig. 3.5 are the frame sub-sequences taken for correlation computation. If VFI_h and VFI_v of any frame sub-sequence are highly correlated (above a threshold, T_{dup}) with that of C, the corresponding sub-sequence is considered to be duplicated between t_{p1} and t_{p2} as C. Algorithm 1 provides the steps involved in 2 t_p frame duplication detection. Using VFI correlation instead of direct frame correlation reduces computation time and is robust to compression.

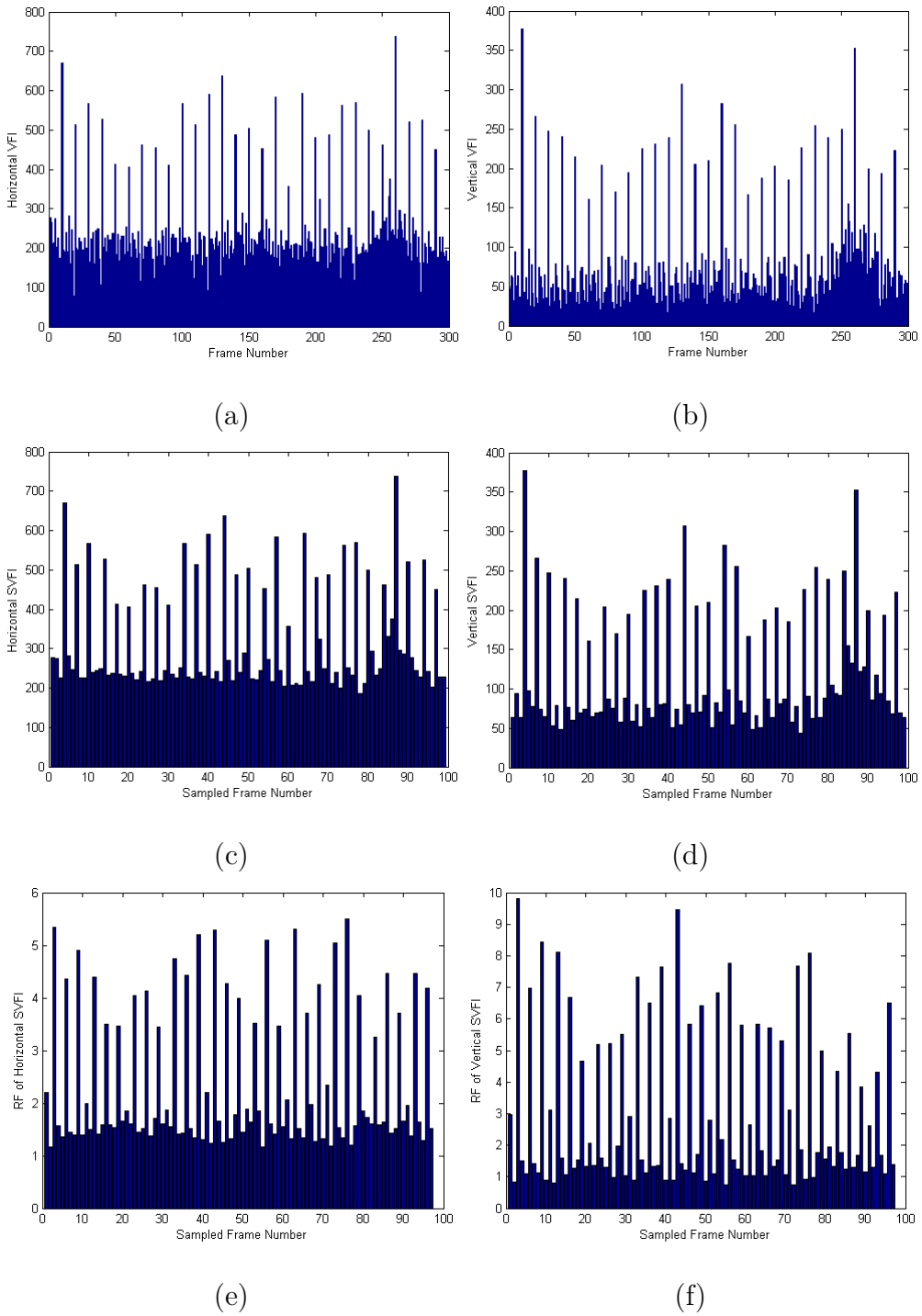


Figure 3.2: VFI, SVFI and RF sequences of pristine video (Container) from [145]. (a) and (b) correspond to VFI_h and VFI_v respectively. (c) and (d) are $SVFI_h$ and $SVFI_v$ computed from (a) and (b) respectively. (e) and (f) are RF_h and RF_v computed from (c) and (d) respectively.

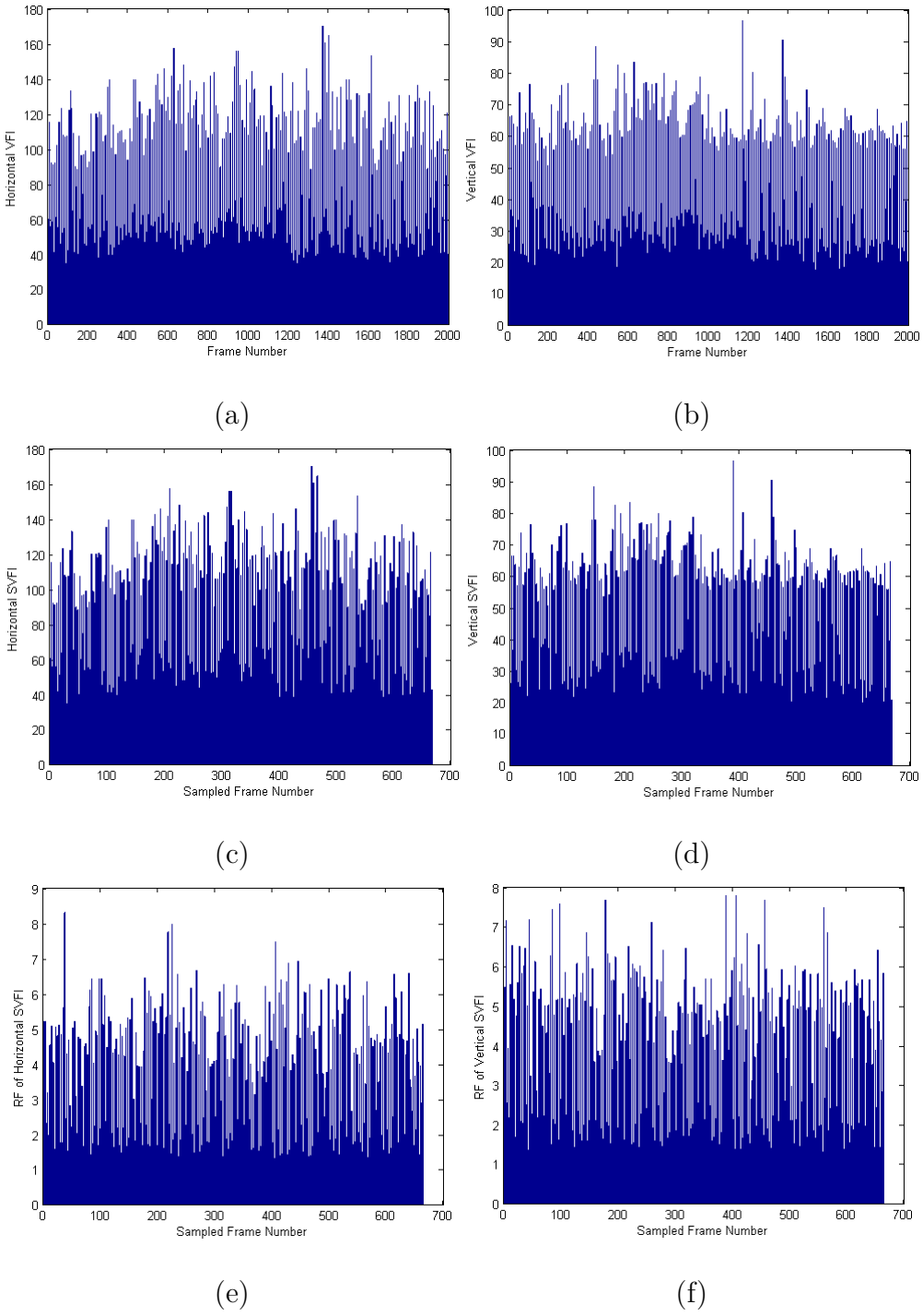


Figure 3.3: VFI, SVFI and RF sequences of pristine video (Bridge-Close) from [145]. (a) and (b) correspond to VFI_h and VFI_v respectively. (c) and (d) are $SVFI_h$ and $SVFI_v$ computed from (a) and (b) respectively. (e) and (f) are RF_h and RF_v computed from (c) and (d) respectively.

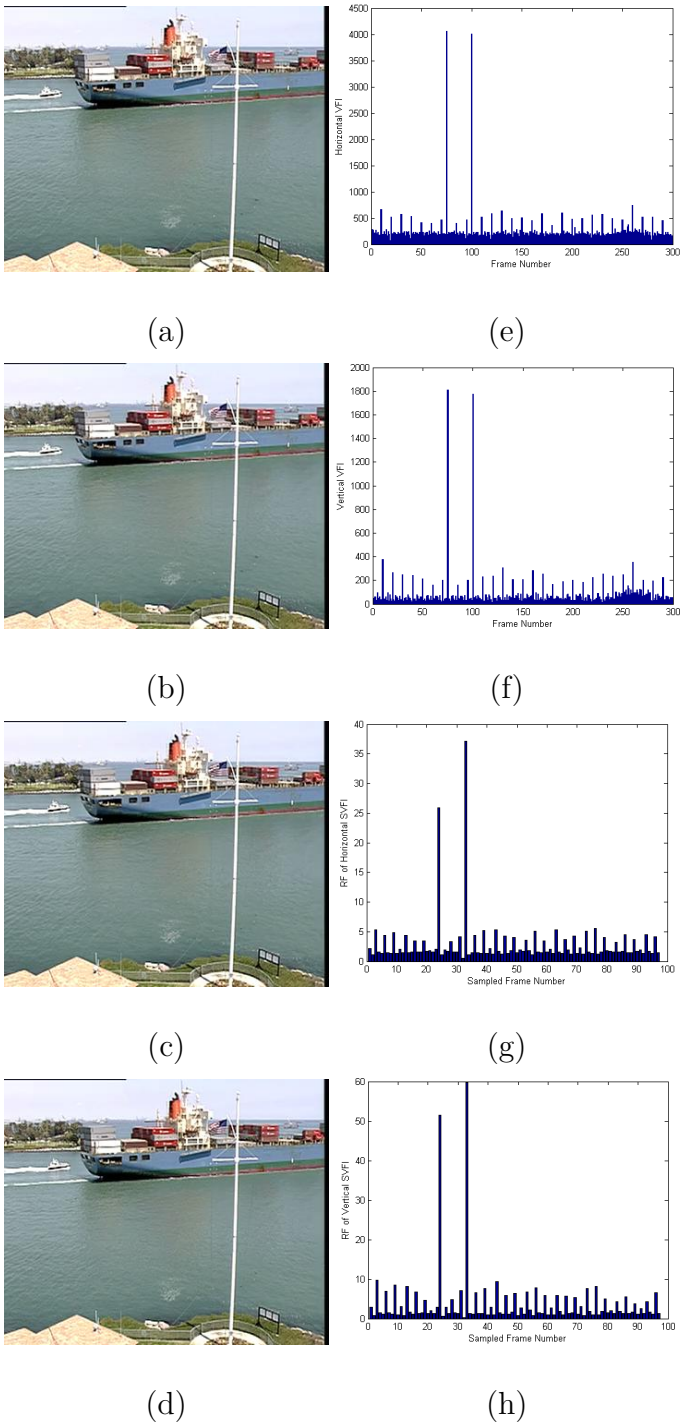


Figure 3.4: Frames of Container video from [145] having 2 t_p after frame duplication. (a), (b), (c), and (d) correspond to frames 75, 76, 100 and 101 respectively. (e) and (f) are VFI_h and VFI_v computed from tampered video. (g) and (h) are RF_h and RF_v computed from the SVFIs of (e) and (f) respectively.

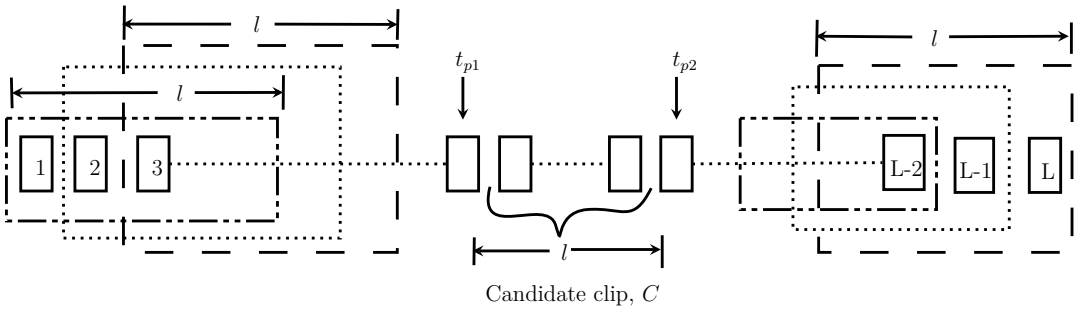


Figure 3.5: Sliding window movement for checking 2 t_p frame duplication

Algorithm 1 Procedure for 2 t_p Duplication Detection

$t1 \leftarrow min(t_{p1}, t_{p2})$; $t2 \leftarrow max(t_{p1}, t_{p2})$;
$l \leftarrow t2 - t1$; $flag_{dup} \leftarrow 0$;
$t\_VFI_h \leftarrow VFI_h$ sub-sequence between $t1$ and $t2$;
$t\_VFI_v \leftarrow VFI_v$ sub-sequence between $t1$ and $t2$;
$ic \leftarrow 0$; $i \leftarrow 1$;
for each sub-sequence of length l starting from i^{th} frame in the video **do**
    if $i < (t1 - l + 1)$ or $i > t2$ **then**
        Assign VFI_h and VFI_v sub-sequences from i to $i + l$ on to s_VFI_h and s_VFI_v respectively;
        $ic \leftarrow ic + 1$;
        Assign correlation between s_VFI_h, and t_VFI_h to $cor_h(ic)$;
        Assign correlation between s_VFI_v, and t_VFI_v to $cor_v(ic)$;
        $frame\_no(ic) \leftarrow i$;
    end if
    $i \leftarrow i + 1$;
end for
$mcor_h \leftarrow max(cor_h)$; $mcor_v \leftarrow max(cor_v)$;
if $mcor_h$ or $mcor_v$ greater than T_{dup} **then**
    $flag_{dup} \leftarrow 1$; /* for indicating successful duplication detection;
    Assign position of $mcor_h$ or $mcor_v$ to p;
    l frames from $frame\_no(p)$ are the original frames used for duplication;
end if

2 t_p Shuffling Detection The tamper points (t_{p1} and t_{p2}) may be the begin-

ning and ending of tampered frames. As the order of replicated frames are rearranged in frame shuffling, VFI sub-sequences of original and duplicated frames will not be similar. If frame duplication returns negative value ($flag_{dup} = 0$), then shuffling test is performed. The frames in between t_{p1} and t_{p2} are regarded as the candidate clip, C for shuffling detection. VFI_h and VFI_v between all frame combinations existing in C are estimated to create two 1-D arrays, C_{hvfi} and C_{vvfi} respectively. This means that the VFIs between the first frame and rest of the frames in C are estimated followed by the VFI computations between the second frame with the rest of the frames in C and so on. C_{hvfi} and C_{vvfi} are then sorted (SC_{hvfi} and SC_{vvfi}) to impart robustness to shuffling. We divide the video into overlapping sub-sequences. The length of each sub-sequence is l which is computed as in 2 t_p duplication. VFI_h and VFI_v between all frame combinations in each sub-sequence are computed. Each of these VFI sub-sequences are then sorted for performing correlation with SC_{hvfi} and SC_{vvfi}. If any of these are highly correlated, ($\geq T_{shuf}$), then the corresponding sub-sequence represent the original frames used for shuffling forgery and the video is classified as forged with frame shuffling. Algorithm 2 and 3 give the procedure for 2 t_p frame shuffling detection. In [102], we tackled frame shuffling by finding the frame correlation of all possible frame combinations in candidate clip and other segments of same size as that of the candidate clip. The correlations obtained between the frame correlation sequences of candidate clip and other segments are used for classifying the video as forged with shuffled frames or not.

Algorithm 2 Procedure for 2 t_p Shuffling Detection

1: $t1 \leftarrow min(t_{p1}, t_{p2})$; $t2 \leftarrow max(t_{p1}, t_{p2})$;
2: $l \leftarrow t2 - t1$; $jc \leftarrow 0$; $flag_{shuf} \leftarrow 0$;
3: $C \leftarrow$ candidate clip between $t1$ and $t2$;
4: **for** j from 1 to l **do**
5: **for** k from 1 to l **do**
6: **if** $j \neq k$ **then**
7: ▷ Page break!

Algorithm 3 Procedure for 2 t_p Shuffling Detection Continued

$jc \leftarrow jc + 1$;
Assign horizontal and vertical VFIs between frames j and k to $C_{hvfi}(jc)$ and $C_{vvfi}(jc)$ respectively;
end if
end for
end for
Sort C_{hvfi} and C_{vvfi} and assign to SC_{hvfi} and SC_{vvfi} respectively;
$ic \leftarrow 0$; $i \leftarrow 1$;
for each sub-sequence of length l starting from i^{th} frame in the video **do**
if $i < (t1 - l + 1)$ or $i > t2$ **then**
$jc \leftarrow 0$;
for j from 1 to l in the sub-sequence **do**
for k from 1 to l in the sub-sequence **do**
if j $\neq$ k **then**
$jc \leftarrow jc + 1$;
Assign horizontal and vertical VFIs between frames j and k to $s_{hvfi}(jc)$ and $s_{vvfi}(jc)$ respectively;
end if
end for
end for
Sort s_{hvfi} and s_{vvfi} and assign to Ss_{hvfi} and Ss_{vvfi} respectively;
$ic \leftarrow ic + 1$;
Assign correlation between Ss_{hvfi}, and SC_{hvfi} to $cor_h(ic)$;
Assign correlation between Ss_{vvfi}, and SC_{vvfi} to $cor_v(ic)$;
$frame\_no(ic) \leftarrow i$;
end if
$i \leftarrow i + 1$;
end for
$mcor_h \leftarrow max(cor_h)$; $mcor_v \leftarrow max(cor_v)$;
if $mcor_h$ or $mcor_v$ greater than T_{shuf} **then**
$flag_{shuf} \leftarrow 1$; /* for indicating successful shuffling detection;
Assign position of $mcor_h$ or $mcor_v$ to p;
l frames from $frame\_no(p)$ are the frames used for shuffling;
end if

2 t_p Insertion Detection If the results of 2 t_p frame shuffling and 1 t_p frame duplication tests indicate that the video under investigation is not forged with either of these ($flag_{dup} = 0$ and $flag_{shuf} = 0$), then it is classified as frame insertion tampering. As $Cardinality(t_p) = 2$, it cannot be frame deletion. The maxmimum number of t_p a frame deletion tampering can contribute is only one. And, it occurs at the temporal location from where a sequence of frames are deleted. Frame insertion usually give rise to 2 tamper points: one at the beginning of insertion and other at the end. For identifying whether the inserted frames have similar background as that of the pristine frames in the video, we compute two frame correlations: (1) frame at t_p1 and the frame that immediately precedes it; and (2) frame at t_p2 and the frame that immediately follows it. If the correlation values obtained are poor, $\leq T_{ins}$, it indicates that the frames used in forgery are having a different background. If not, the inserted frames have similar FOV and may be taken from a video captured on some other time or day or by using another camera with almost same FOV.

Case 2 There is a single abnormal peak (t_p) in the RF sequences. Such videos are classified as tampered. It may be because of frame duplication, shuffling, or deletion. The chances of performing frame insertion at either end of video are less as it may be visually noticeable. Figure 3.6 shows VFI and RF distributions with one tamper point. Frames 850 to 1890 are deleted from Bridge-Close video [145]. 850^{th} frame in tampered video is 1891^{th} frame in pristine video which is 1042 frames away from the 849^{th} frame. This time gap between the new neighboring frames in the tampered will disturb the distribution of VFI sequences. VFI and RF sequences of tampered video in Fig. 3.6 (b), (e), (c) and (f) can be compared with those of its pristine video from Fig. 3.3 (a), (b), (e) and (f) respectively.

1 t_p Duplication Detection There are 3 ways through which this case can occur: (1) copied frames are placed at the beginning of a video (2) copied frames are placed after the last frame in a video or (3) method used for identifying t_p in a video fails in detecting one of them because of plausible editing. The length of candidate clip is unknown. The direction from t_p,

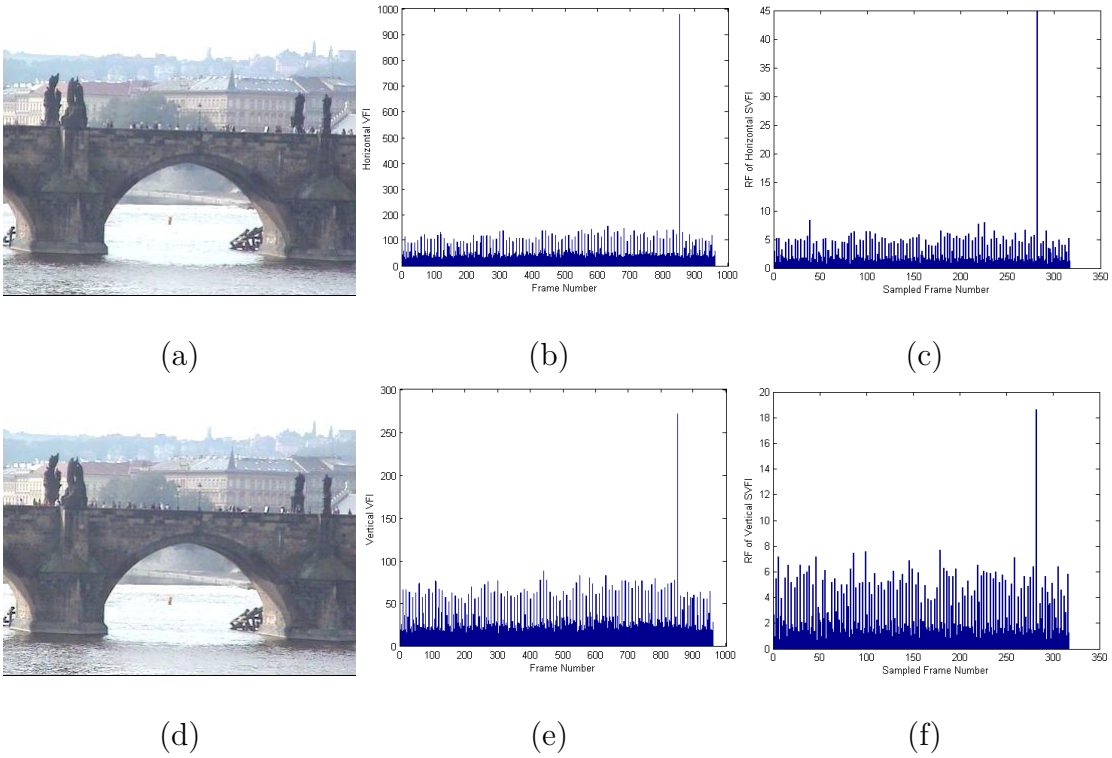


Figure 3.6: Frame of Bridge-Close video from [145] having 1 t_p after frame deletion. (a) and (d) are the 849^{th} and 850^{th} frames respectively of the tampered video. (b) and (e) are VFI_h and VFI_v computed from this tampered video. (c) and (f) are RF_h and RF_v computed from the SVFIs of (b) and (e) respectively.

the copied frames are placed is also unknown. To find the candidate clip, l frames preceding t_p are grouped to form a sample candidate clip C_1 and those following t_p form C_2 where $l = fps$. Like in Case 1 duplication detection, VFI sub-sequences of C_1 and C_2 are correlated with that from all other frame sub-sequences. The frame sub-sequence which is highly correlated to either C_1 or C_2 and having a correlation value above threshold T_{dup} is taken as the original frames used for duplication forgery. The length of tampering may be more than fps. For finding the entire frames involved in forgery, this process has to be repeated either in the direction of C_1 or towards C_2. The direction is chosen based on for which candidate clip (C_1 or C_2), the duplicates are found. If C_1, then l frames preceding C_1 are taken

for updating the current C_1. Duplication check is performed by correlating the VFI sequences of l frames that precede the sub-sequence which were highly correlated with the previous C_1 with current C_1. If C_2, then l frames following C_2 are taken for updating C_2. Duplication check is performed with l frames that follow the sub-sequence which matched the previous C_2. This process of updating the candidate clip and checking for duplication continues until VFI correlation is disturbed. The 1^{st} sub-sequence having low VFI correlation is divided into sub-groups of equal length ($l/2$) for checking duplication. C_1 or C_2 have to be divided accordingly. Figure 3.7 gives a pictorial representation of the method. If duplicated frames are on the left of t_p, then we continue duplication check with right sub-group (l_r). If VFI of this l_r is highly correlated with that of the corresponding sub-group of C_1, then l_l is split into two halves (l_{ll} and l_{lr}). l_{lr} forms the candidate clip for checking duplication. This process of splitting and checking continues until the length of a sub-group is less than 5. If VFI of l_r is not correlated, then l_r is split into l_{rl} and l_{rr}. l_{rr} is taken for testing. Likewise, if duplicated frames are on the right of t_p then left sub-group (r_l) is taken for further testing. If r_l is highly correlated to current C_2, then r_r is split into r_{rl} and r_{rr}. Otherwise r_l is split into r_{ll} and r_{lr}. Frames sub-sequences present in first candidate clip to the last one form the pristine frames used in duplication forgery at t_p. Algorithm 4 and 5 provide the procedure for 1 t_p duplication detection. Algorithms 6 and 7 are its sub-procedures.

1 t_p Shuffling Detection If 1 t_p frame duplication detection method returns negative result ($flag_{dup} = 0$), then the video under investigation is analyzed for frame shuffling forgery. We assume that the shuffled frames are placed either at the beginning or the end of a video. As the method presented in this chapter is an initial work on frame shuffling detection and therefore it does not address plausible video editing that removes single tamper points (shuffled frames placed anywhere in the video). The feature used to identify C is frame count. The frame count on either side of t_p is computed. The frame sequence having lower frame count is considered as C. With C, the method discussed for 2 t_p shuffling detection is used.

Algorithm 4 Procedure for 1 t_p Duplication Detection

1: $l \leftarrow fps$; $flag_{dup} \leftarrow 0$; $ic_1 \leftarrow 0$; $ic_2 \leftarrow 0$; $i \leftarrow 1$;
2: Assign VFI sub-sequences between $t_p - l$ and t_p to C_{1hvfi} and C_{1vvfi};
3: Assign VFI sub-sequences between t_p and $t_p + l$ to C_{2hvfi} and C_{2vvfi};
4: **for** each sub-sequence of length l starting from i^{th} frame in the video **do**
5: **if** $i < (t_p - l + 1)$ or $i > t_p$ **then**
6: Assign VFI sub-sequences from i to $i + l$ on to s_{hvfi} and s_{vvfi};
7: $ic_1 \leftarrow ic_1 + 1$;
8: Assign correlation between s_{hvfi}, and C_{1hvfi} to $cor1_h(ic_1)$;
9: Assign correlation between s_{vvfi}, and C_{1vvfi} to $cor1_v(ic_1)$;
10: $frame1\_no(ic_1) \leftarrow i$;
11: **end if**
12: **if** $i < t_p$ or $i > (t_p + l)$ **then**
13: Assign VFI sub-sequences from i to $i + l$ on to s_{hvfi} and s_{vvfi};
14: $ic_2 \leftarrow ic_2 + 1$;
15: Assign correlation between s_{hvfi}, and C_{2hvfi} to $cor2_h(ic_2)$;
16: Assign correlation between s_{vvfi}, and C_{2vvfi} to $cor2_v(ic_2)$;
17: $frame2\_no(ic_2) \leftarrow i$;
18: **end if**
19: $i \leftarrow i + 1$;
20: **end for**
21: $mcor1_h \leftarrow max(cor1_h)$; $mcor1_v \leftarrow max(cor1_v)$;
22: $mcor2_h \leftarrow max(cor2_h)$; $mcor2_v \leftarrow max(cor2_v)$;
23: $mcor_h \leftarrow max(mcor1_h, mcor2_h)$; $mcor_v \leftarrow max(mcor1_v, mcor2_v)$;
24: $flag \leftarrow 1$; $i \leftarrow 1$;
25: **if** $mcor_h = mcor1_h$ and $mcor_v = mcor1_v$ **then**
26: Assign position of $mcor1_h$ or $mcor1_v$ to p;
27: **if** $mcor_h$ or $mcor_v$ greater than T_{dup} **then**
28: $flag_{dup} \leftarrow 1$; /* for indicating successful duplication detection
29: **if** flag **then**
30: $t1 \leftarrow frame1\_no(p)$; $t2 \leftarrow frame1\_no(p) + l$; $flag \leftarrow 0$;
31: **else**
32: $t1 \leftarrow t2 - (i \cdot l)$;
33: **end if**
34: ▷ Page break!

Algorithm 5 Procedure for 1 t_p Duplication Detection continued

Update $C1$ with l frames from $t_p - (i \cdot l)$;
$i \leftarrow i + 1$;
Update s_{hvfi} and s_{vvfi} with VFI sub-sequences of length l from $t2 - (i \cdot l)$;
Compute $mcor_h$ and $mcor_v$ by correlating s_{hvfi} and s_{vvfi} with new C_{1hvfi} and C_{1vvfi} respectively and go to step 27;
else
Perform procedure for C_1 clip partitioning and assign the return value to $t1$;
end if
else
Assign position of $mcor2_h$ or $mcor2_v$ to p;
if $mcor_h$ or $mcor_v$ greater than T_{dup} **then**
$flag_{dup} \leftarrow 1$; /* for indicating successful duplication detection
if flag **then**
$t1 \leftarrow frame1\_no(p)$; $t2 \leftarrow frame1\_no(p) + l$; $flag \leftarrow 0$;
else
$t2 \leftarrow t1 + (i \cdot l)$;
end if
Update $C2$ with l frames from $t_p + (i \cdot l)$;
Update s_{hvfi} and s_{vvfi} with VFI sub-sequences of length l from $t1+(i \cdot l)$;
$i \leftarrow i + 1$;
Compute $mcor_h$ and $mcor_v$ by correlating s_{hvfi} and s_{vvfi} with new C_{2hvfi} and C_{2vvfi} respectively and go to step 43;
else
Perform procedure for C_2 clip partitioning and assign the return value to $t2$;
end if
end if
if $flag_{dup}$ **then**
Frames from $t1$ to $t2$ are the original frames used for duplication;
end if

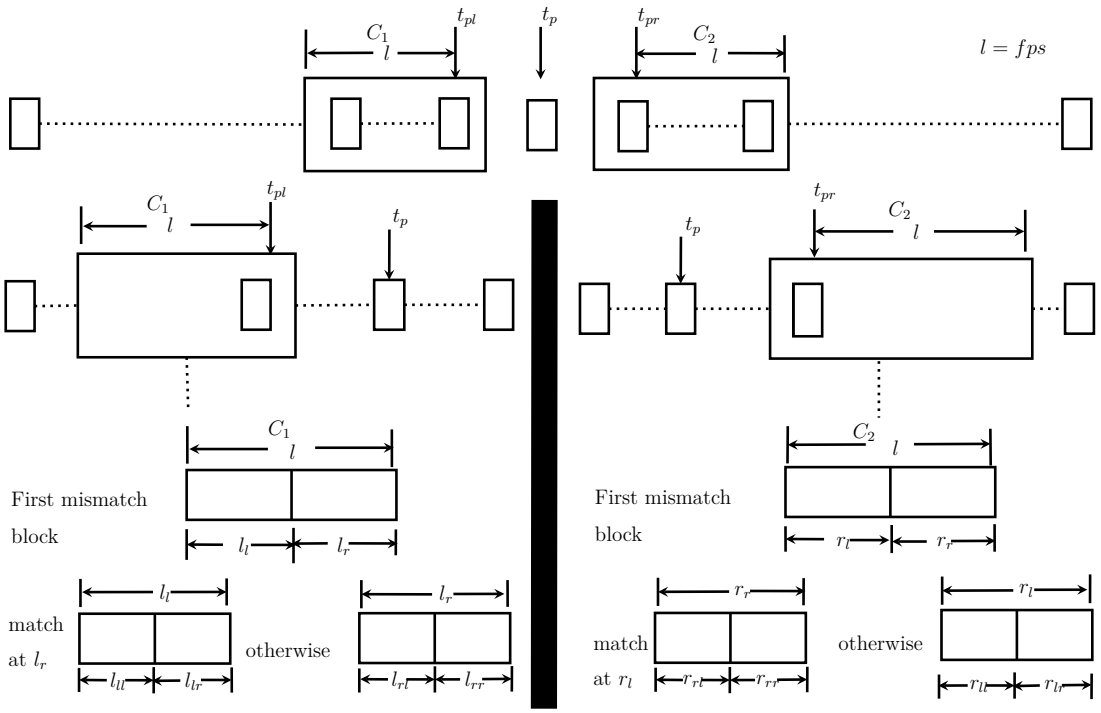


Figure 3.7: Sliding window movement for checking 1 t_p frame duplication

Deletion Detection If the results of 1 t_p frame shuffling and 1 t_p frame duplication tests indicate that the video under investigation is not forged with either of these, then it is classified as frame deletion tampering. As shown in Fig. 3.6, frame deletion brings two temporally distinct frames together causing the distruption of VFI at that point. This leads to a single tamper point.

Case 3 The absence of abnormal peaks in RF sequences implies that the velocity field is consistent. Hence, such videos are categorized as pristine.

Sudden zooming may add irregularities in VPA and VFI. The beginning and ending frames of zooming will have abrupt changes as opposed to the frames that immediately precedes and follows them. Such videos may get wrongly classified as tampered (frame insertion). To reduce such false positives, we discuss a method for zooming detection in the following subsection 3.2.4. The frames that immediately precedes and follows t_p are taken for zooming detection.

Algorithm 6 Procedure for partitioning clip $C1$ in 1 t_p Duplication Detection

1: Partition C into sub-groups Cl and Cr;
2: Partition the corresponding mismatched sub-sequence S into sub-groups Sl and Sr;
3: Compute $mcor_h$ and $mcor_v$ by correlating VFI sequences of Cr and Sr;
4: $flag \leftarrow 1$;
5: **if** $mcor_h$ or $mcor_v$ greater than T_{dup} **then**
6: $l \leftarrow l/2$;
7: **if** $flag$ **then** $t1 \leftarrow t1 - l$;
8: **end if**
9: Partition Cl into sub-groups Cl_l and Cl_r;
10: Partition Sl into sub-groups Sl_l and Sl_r;
11: **if** length of $Cl > 5$ **then**
12: Compute $mcor_h$ and $mcor_v$ by correlating VFIs of Cl_r and Sl_r;
13: **if** $mcor_h$ or $mcor_v$ greater than T_{dup} **then**
14: Update Cl and Sl with Cl_l and Sl_l respectively
15: $flag \leftarrow 1$; goto step 6;
16: **else**
17: Update Cl and Sl with Cl_r and Sl_r respectively;
18: $flag \leftarrow 0$; goto step 6;
19: **end if**
20: **end if**
21: **else**
22: Update Cl and Sl with Cr and Sr respectively;
23: $flag \leftarrow 0$; goto step 6;
24: **end if**
25: return $t1$;

3.2.4 Zooming Detection

We employ a key point based motion vector method for zooming detection. SIFT [68] key points are obtained from consecutive frames at t_p. SIFT key points from two consecutive frames (n^{th} and $(n+1)^{th}$) are matched to determine the motion vectors in these frames. If the position of a matched key point in $(n+1)^{th}$ frame

Algorithm 7 Procedure for partitioning clip $C2$ in 1 t_p Duplication Detection

1: Partition C into sub-groups Cl and Cr;
2: Partition the corresponding mismatched sub-sequence S into sub-groups Sl and Sr;
3: Compute $mcor_h$ and $mcor_v$ by correlating VFI sequences of Cl and Sl;
4: $flag \leftarrow 1$;
5: **if** $mcor_h$ or $mcor_v$ greater than T_{dup} **then**
6: $l \leftarrow l/2$;
7: **if** $flag$ **then** $t2 \leftarrow t2 + l$;
8: **end if**
9: Partition Cr into sub-groups Cr_l and Cr_r;
10: Partition Sr into sub-groups Sr_l and Sr_r;
11: **if** length of $Cr > 5$ **then**
12: Compute $mcor_h$ and $mcor_v$ by correlating VFIs of Cr_l and Sr_l;
13: **if** $mcor_h$ or $mcor_v$ greater than T_{dup} **then**
14: Update Cr and Sr with Cr_r and Sr_r respectively;
15: $flag \leftarrow 1$; goto step 6;
16: **else**
17: Update Cr and Sr with Cr_l and Sr_l respectively;
18: $flag \leftarrow 0$; goto step 6;
19: **end if**
20: **end if**
21: **else**
22: Update Cr and Sr with Cl and Sl respectively;
23: $flag \leftarrow 0$; goto step 6;
24: **end if**
25: return $t2$;

is not in a radius of 30 corresponding to its matching key point's position in n, then they are dropped from motion vector computations. This occurs because of false matches. The span of a frame is 1/25sec if the fps of a video is 25. Hence, zooming operation usually will not create large MVs between consecutive frames. Also, we exclude those matches having a displacement of less than 0.4 as these

may be arising due to the presence of noise in the video.

The motion vector, $\overrightarrow{V} = \langle v_x, v_y \rangle$, is the directed line segment from a key point in nth frame, $A = (x, y)$, to its matched key point in $(n+1)^{th}$ frame, $B(x', y')$ where $v_x = x' - x$ and $v_y = y' - y$. Normally, the motion vectors in a 2-D plane appear as presented in Fig. 3.8(c), X and Y-axes show horizontal and vertical displacements respectively. The normal frames used in the computation of these motion vectors are also presented in Fig. 3.8 (a) and (b). These are two consecutive frames taken from one of the videos we recorded. Zoom-in and zoom-out cause the MVs to appear in a particular pattern due to the divergence and convergence of MVs respectively. The frames and MVs corresponding to zoom-in and zoom-out operations are presented in Fig. 3.9 and 3.10 respectively. MVs in Fig. 3.8(c), 3.9(c) and 3.10(c) are multiplied by a value of 10 to make them visible in these figures.

In a frame, the orientation of MVs are computed:

$$o_i^n = \tan^{-1}\left(\frac{y_i^{(n+1)} - y_i^n}{x_i^{(n+1)} - x_i^n}\right) \tag{3.12}$$

where y_i^n and $y_i^{(n+1)}$ denote the Y co-ordinates of i^{th} matched key point in $(n+1)^{th}$ and n^{th} frames respectively. Similarly, x_i^n and $x_i^{(n+1)}$ denote the X coordinates of the same key point. o^n can take values in $[-180°, 180°]$. These are split into sectors of 45°. An 8 bin orientation histogram of angles are computed from this. 8 bin orientation histograms obtained for normal, zoom-in and zoom-out MVs in Fig. 3.8 (c), 3.9 (c) and 3.10 (c) are presented in Fig. 3.11 (a), (b) and (c) respectively. These figures reveal that for a normal frame, most of its angles are ≈ 0. And in case of a zoomed frame, the angles are distributed amongst all the bins. It is because of divergence or convergence of MVs. Figure 3.12 presents MVs computed from another normal frame containing visuals of random movement due to waving leaves. The orientation of MVs in Fig.3.12 (c) is scattered like in Fig. 3.8 (c). The histogram estimated is normalized so that the area under the probability density function is equal to 1. The normalized histograms obtained from Fig. 3.11 (a), Fig.3.12 (c), Fig. 3.11 (b) and (c) are presented in Fig. 3.13 (a), (b), (c) and (d) respectively. The number of bins in the normalized histograms of frames without zooming whose values are ~ 0 is high. The highest peak bin may

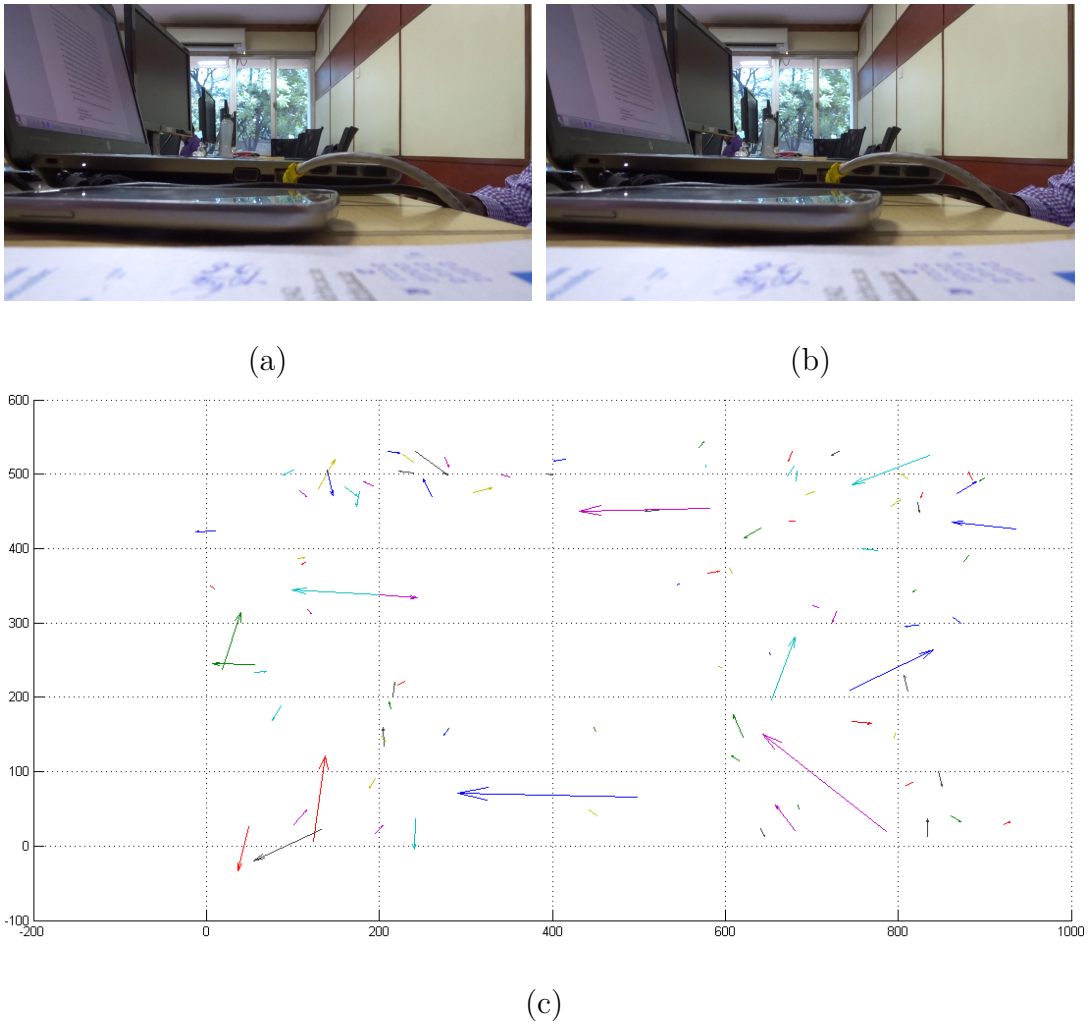


Figure 3.8: Motion vectors of a normal video. (a) and (b) are two consecutive frames of a normal video without zooming. (c) Motion vectors computed from (a) and (b), X and Y axes show the displacements along horizontal and vertical directions respectively.

be the one that corresponds to 0° which indicates the absence of moving objects or some other bin indicating camera pan or object movement in that direction. This distribution of angles is used as a feature for detecting videos with zoomed frames. If the count of bins having frequency values less than a threshold, T_{z1} is less than or equal to T_{z2}, then that frame is classified as zoomed. The value of

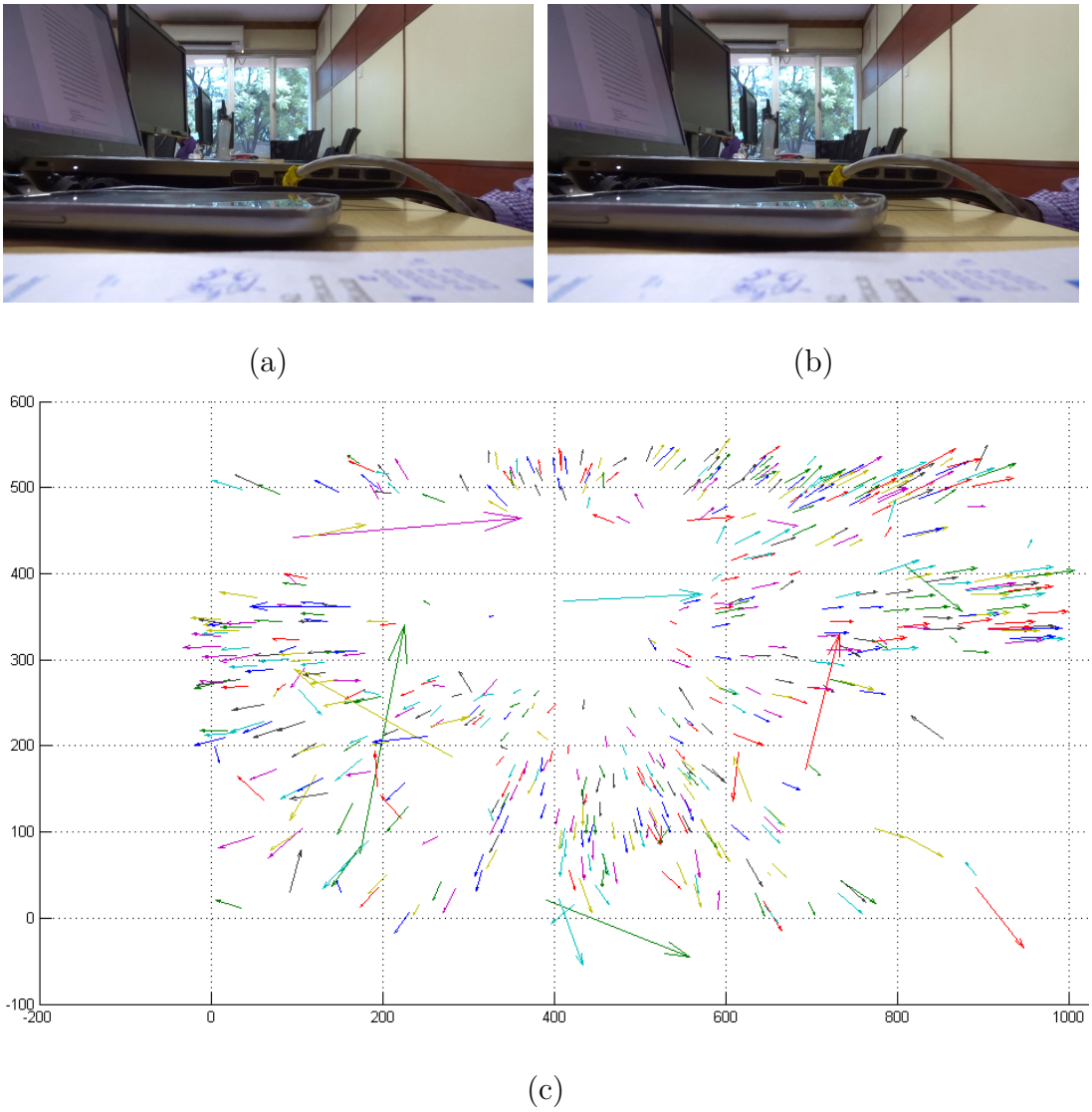


Figure 3.9: Divergence of motion vectors during zoom-in. (a) and (b) are two consecutive frames recorded during zoom-in. (c) Motion vectors computed from (a) and (b).

T_{z1} is set 0.04 experimentally. It makes the frequency of angles in a sector ≈ 0. The value of T_{z2} is set 3 experimentally. T_{z2} should be less than 4 (1/2 of the number of bins in the histogram) which indicates that angles are distributed.

To reduce the false positives due to moving objects, the divergence or conver-

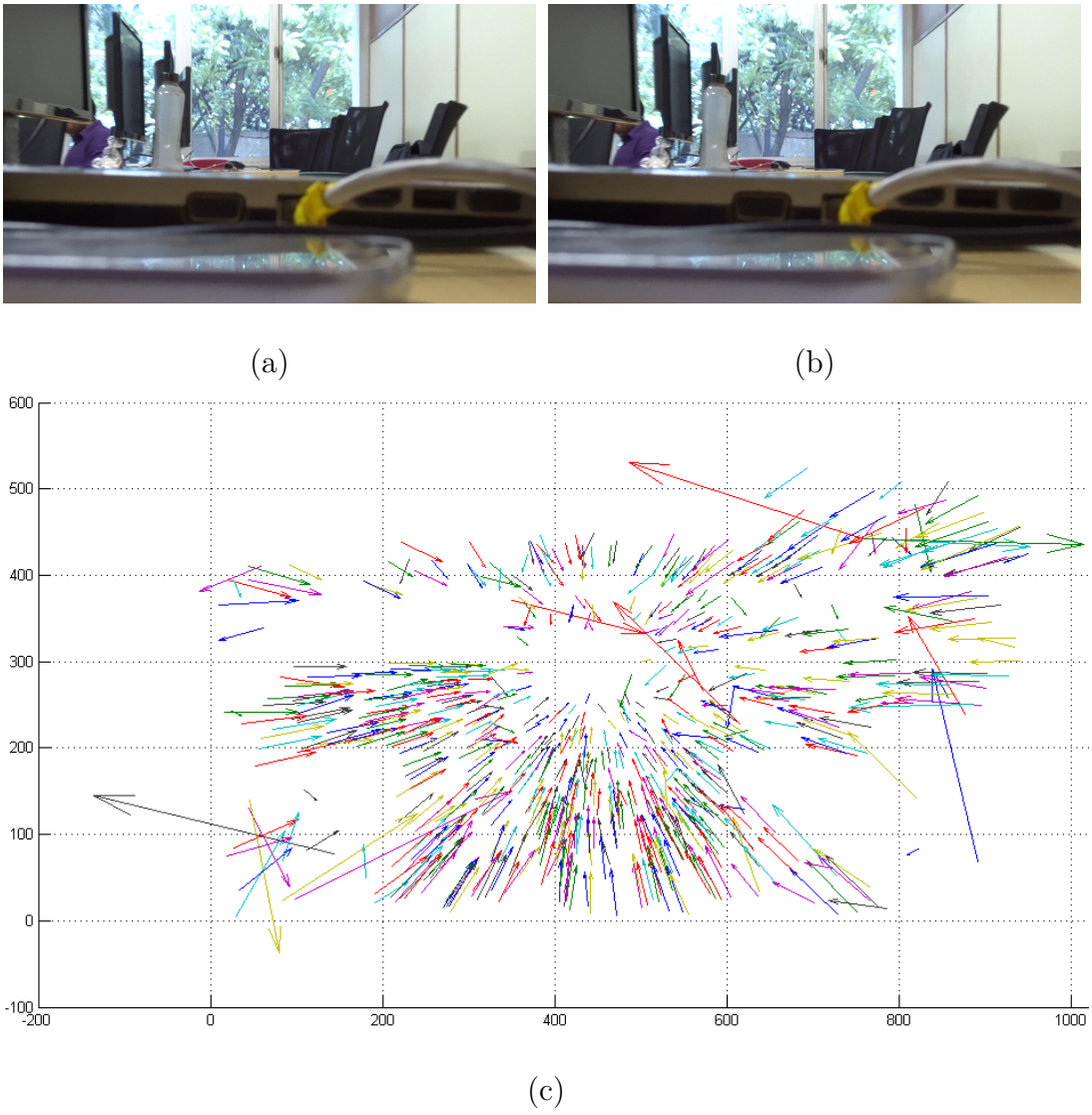
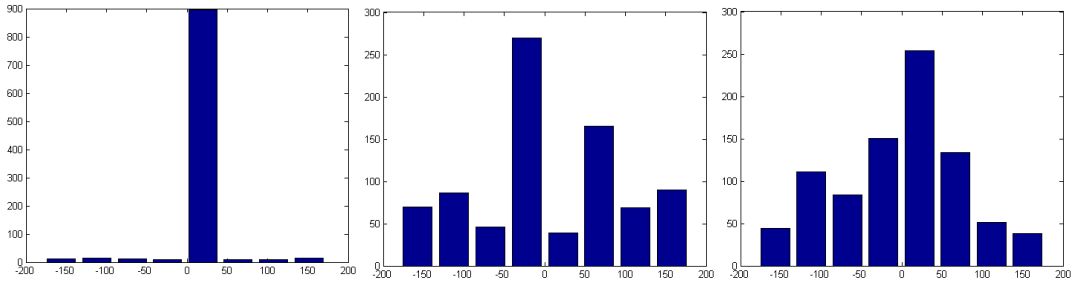


Figure 3.10: Convergence of motion vectors during zoom-out. (a) and (b) are two consecutive frames recorded during zoom-out. (c) Motion vectors computed from (a) and (b).

gence of MVs are evaluated. Zoom-out causes (x_i^{n+1}, y_i^{n+1}) to be nearly closer to the frame center, (x_c, y_c), than (x_i^n, y_i^n). Similarly, zoom-in causes (x_i^n, y_i^n) to be nearly closer to (x_c, y_c), than (x_i^{n+1}, y_i^{n+1}). We employ Euclidean distance to find the distances between (x_i^n, y_i^n) and (x_i^{n+1}, y_i^{n+1}) from (x_c, y_c) for classifying the



(a) (b) (c)

Figure 3.11: 8 bin orientation histogram of angles. (a) Normal frame, (b) Zoom-in frame, and (c) Zoom-out frame.

zooming type:

$$\sqrt{(x_i^{n+1} - x_c)^2 + (y_i^{n+1} - y_c)^2} < \sqrt{(x_i^t - x_c)^2 + (y_i^t - y_c)^2} \tag{3.13}$$

If Eq. 3.13 is satisfied, then it indicates that MV is pointing inwards (convergence). If not, it conveys that MV is pointing outwards (divergence). The dominance of divergence or convergence of MVs is taken into account to classify the frame as zoomed-in or zoomed-out. We compute the percentage of MVs which are diverging and converging. If this percentage of divergence or convergence is $\approx 90\%$, we classify the frames as zoomed-in or zoomed-out. If neither of this conditions follows, then it is classified as a normal frame.

3.3 Experimental Results and Discussion

The proposed method is implemented using MATLAB R2013a (8.1.0.604), performed on 3.40 GHz Intel Core i7 PC. For parameter tuning and performance evaluation of the proposed method, the inter-frame video tamper detection dataset discussed in Subsection 2.7.2 of Chapter 2 is used. All video frames are converted to gray scale before processing it for inter-frame video tampering detection. Here, the description of TP, FN, TN and FP are as follows when video tampering is considered as the positive class:

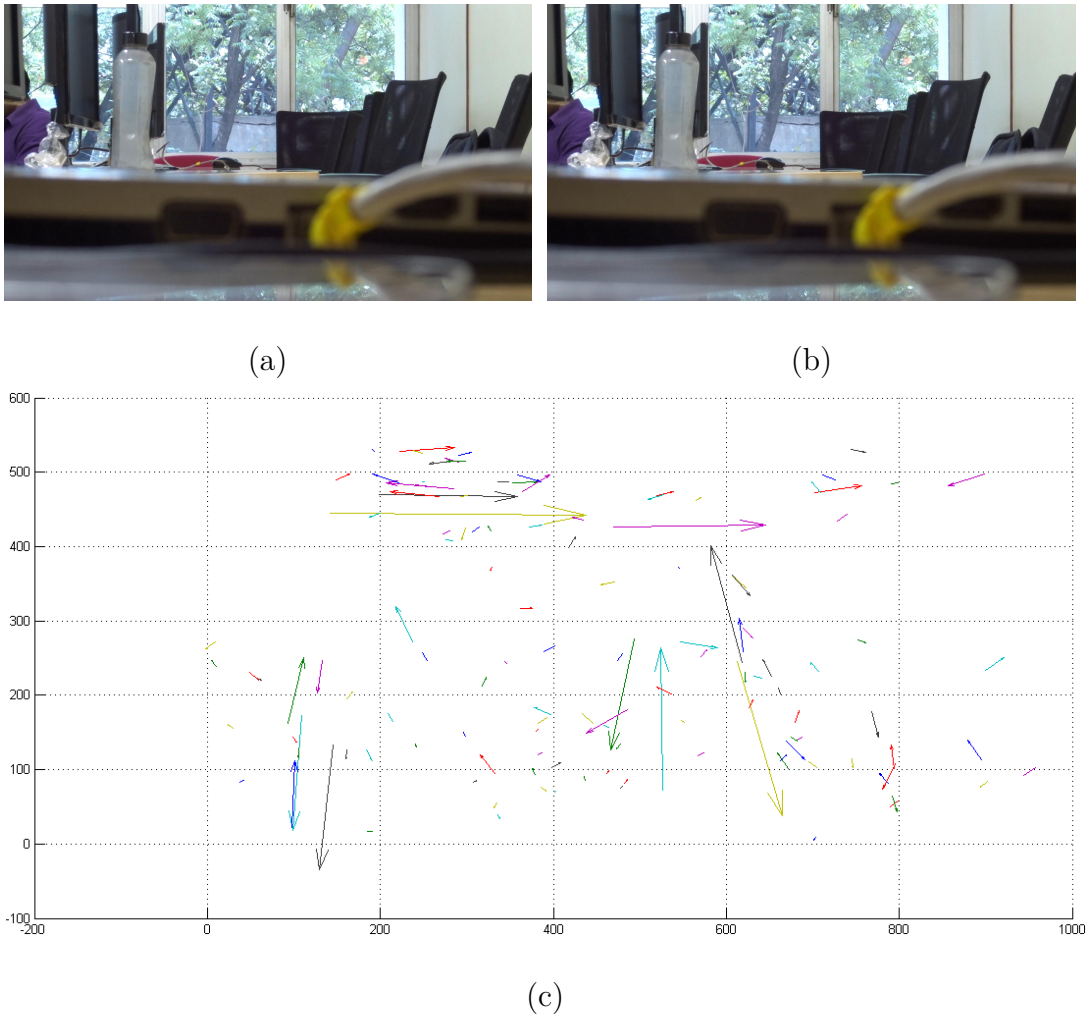


Figure 3.12: Motion vectors of a normal video containing visuals of waving leaves. (a) and (b) are two consecutive frames of a normal video without zooming. (c) Motion vectors between (a) and (b).

- **TP** - A tampered video given to the proposed method for forgery detection classified as tampered and identified the correct inter-frame forgery type.
- **TN** - A pristine video classified as pristine.
- **FP** - A pristine video classified as tampered.
- **FN** - A tampered video detected as pristine or failed in identifying the

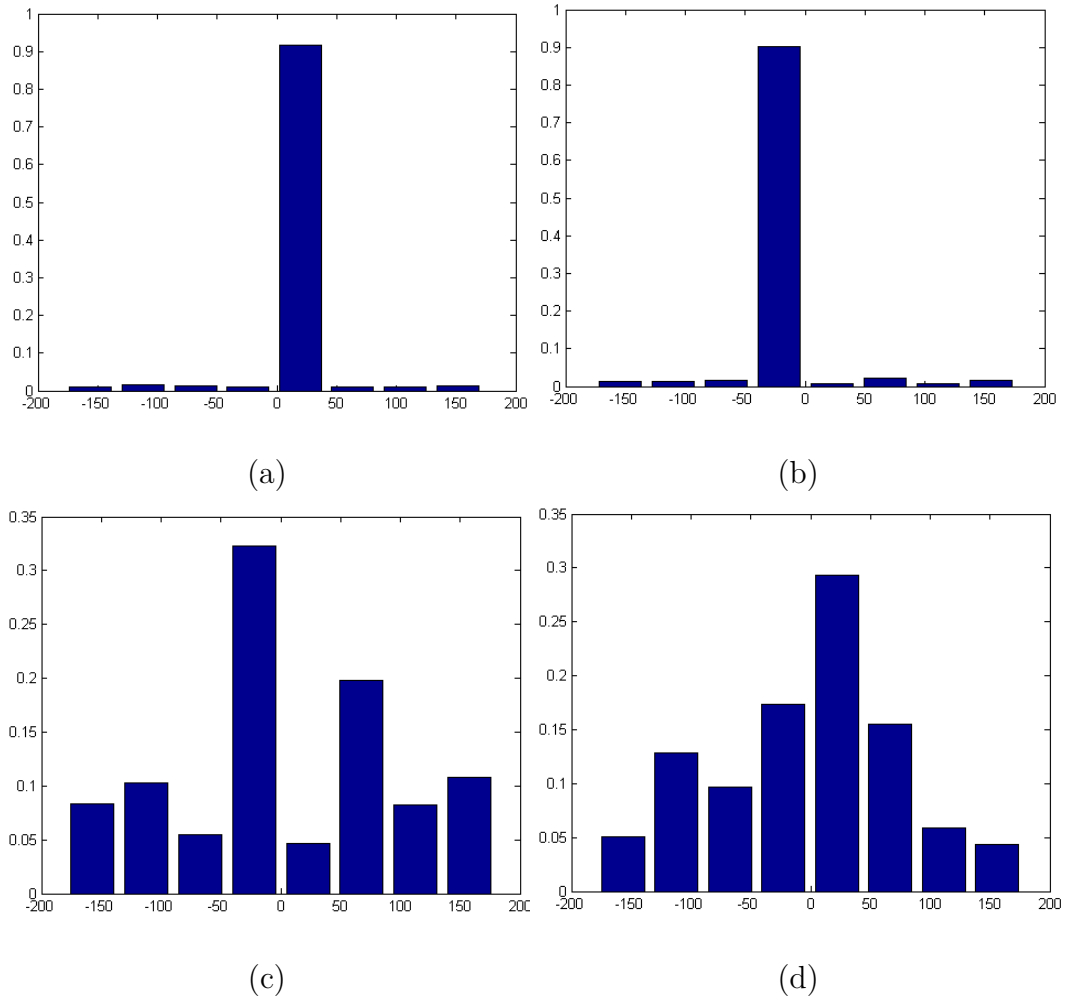


Figure 3.13: Normalized Histogram. (a) Normal frame, (b) Normal frame with waving leaves, (c) Zoom-in frame, and (d) Zoom-out frame.

correct inter-frame forgery type.

3.3.1 Inter-frame Video Tampering Dataset Created

Public datasets dedicated to inter-frame video tampering detection are not available. We have created our own dataset from pristine video datasets discussed in Subsection 2.7.2 of Chapter 2.

3.3.1.1 Video Dataset used

Bridge-Close, Mobile, Akiyo, Highway, Waterfall, Coastguard, Silent, Carphone, Galleon, Mother and Daughter, Claire, Tempete, Container, Sign-Irene, News, Foreman, Pamphlet, Grandma, Hall, Husky, Paris, Bowing, and Bridge-Far are the videos downloaded from [118, 145]. These videos are in YUV uncompressed format with CIF (352×288) resolution. The resolution of videos from PETS2007, PETS2009 and PETS2001 are 720×576, 768×576, and 768×576 respectively. Adding to this, we recorded 35 videos of HD resolution.

3.3.1.2 Parameters and Codecs used for Compression

In order to perform inter-frame video tampering, a compressed video has to be decompressed. After the tampering activity, the videos need to be stored back in a compressed format. Therefore, a tampered video will experience at least two levels of compression. We compressed videos in MPEG-4 and H.264 compression formats using libavcodec and libx264 libraries of ffmpeg [117] with AGOP structure. Videos encoded with MPEG-4 prior editing are recompressed using H.264 or MPEG-4 post editing. Similarly, H.264 encoded videos are recompressed using H.264 or MPEG-4 post editing. Tampered sequences are created using Constant Bit Rate (CBR) and Variable Bit Rate (VBR) coding for examining the accuracy of proposed methods under different rate control modes. The bitrate of a video encoded in CBR mode remains constant. Hence, it is mostly used in band-limited communications. If bitrate is low, it may reduce the visual quality of a video depending on its resolution. In VBR coding, rate factor implies the quality expected for a video. Hence, it varies bitrates accordingly. For CBR coding with H.264 and MPEG-4, bit rates were chosen from .2, .4, .7 Mbps for video sequences from mediaxiph and 1, 2, 3 Mbps for video sequences from PETS2001, PETS2007 and PETS2009 because of the difference in resolution. These bitrates can be denoted as b1, b2, b3. For VBR coding with MPEG-4, the value of Qscale is taken from 3, 5, 9. The value of crf (constant rate factor) is taken from 18, 23, 28 for VBR coding with H.264. These VBR factors can be denoted as c1, c2, c3. The encoders and parameters taken for creating tampered videos are given in Table 3.1.

Table 3.1: Compression parameters used for creating pristine and tampered videos in first and second encoding

Parameters	**1st encoding**	**2nd encoding**
Encoder	{MPEG-4, H.264}	{MPEG-4, H.264}
Bitrate (Mbps)	$\{.2, .4, .7\}$	$\{.2, .4, .7\}$
	$\{1, 2, 3\}$	$\{1, 2, 3\}$
Qscale (MPEG-4)	$\{3, 5, 9\}$	$\{3, 5, 9\}$
crf (H.264)	$\{18, 23, 28\}$	$\{18, 23, 28\}$

3.3.1.3 Video Dataset Creation

90 video sequences were chosen from the datasets mentioned in Subsection 3.3.1.1 for producing inter-frame tampered videos with the following tamper type: insertion, duplication, and deletion. The length of tamper sequence in fabricated videos ranges from 25 to 1000 frames. Here, we considered convincing inter-frame forgeries, i.e., the continuance of an event is at least 1 sec. Therefore, the minimum length (number of frames) of manipulation is 25. For every tamper type, videos with VBR and CBR coding are generated with their corresponding parameters for H.264 and MPEG-4. This sum up to 6480 tampered videos in each tamper type (insertion, duplication, and deletion). Half of the videos in these classes are CBR coded while the rest is VBR coded.

Videos from static surveillance systems are chosen for creating forged videos with frame shuffling. 7 and 6 videos comprising static frames with no movement are selected from PETS2009 and PETS2001 respectively. We recorded 15 static videos of HD resolution. We created a total of 2016 shuffled videos in VBR and CBR coding.

We recorded 20 videos containing zoom-in and zoom-out frames for our experiments with zooming detection method. We chose 5, 10, and 5 normal video sequences from PETS2007, PETS2009, and PETS2001 respectively, totaling 20

videos. The visual contents in these videos include static frames, people, and vehicles. Additionally, 2 videos with zooming (Waterfall and Tempete) are chosen from [145]. This comprises to 20 normal and 22 zoomed videos for zooming detection.

We take 20 videos with zoom-in and zoom-out frames recorded for zooming test for inter-frame video tampering detection also. 15 videos recorded by us for detecting frame shuffling; 23 from [118, 145]; 104, 20, and 16 videos from PETS2009 (S0, S1 and S2 subsets), PETS2001 and PETS2007 are also used. This collectively forms 198 pristine videos. When compressed in different codecs and parameters for the first compression, it totals to 2376 videos.

Table 3.2 provides information about videos taken from the datasets in Subsection 3.3.1.1 for generating pristine and tampered videos. The third last column (Video Recorded) indicates the number of videos that we recorded for the detection of frame shuffling and zooming. We also consider them for assessing the accuracy of the proposed system in classifying pristine videos. As we attempted to produce convincing forgery with the minimal visual distortion between adjacent frames at t_p, the number of videos chosen from PETS2009 and PETS2007 for creating duplication, deletion, insertion, and shuffling forgeries are different from the videos in the pristine category. The column corresponding to "Total Videos" provides the count of distinct videos in each category. The column corresponding to "Compressed Videos" provides the total number of compressed videos following VBR and CBR coding in each class. Corresponding to each tampered video, there exists 36 CBR, and 36 VBR compressed videos generated using different combinations of the compression factors provided in Subsection 3.3.1.2 for the first and second compressions. For each video in the pristine category, there exists 6 CBR, and 6 VBR compressed videos generated using different combinations of compression factors for the first compression. We randomly selected 22 zoomed videos from the pristine compressed category for testing the proposed method for zooming detection.

Table 3.2: Summary of videos taken from the pristine video datasets (Trace [118, 145], PETS2001 [79], PETS2007 [26] and PETS2009 [25]) used for creating different versions of pristine and tampered videos

Video Type	PETS 2001	PETS 2007	PETS 2009	Trace	Video Recorded	Total Videos	Compressed Videos
Pristine	20	16	104	23	35	198	2376
Deletion	20	4	43	23	–	90	6480
Insertion	20	4	43	23	–	90	6480
Duplication	20	4	43	23	–	90	6480
Shuffling	6	–	7	–	15	28	2016
Zooming	–	–	–	2	20	22	22

3.3.2 Parameter Tuning

T_{shuf}, T_{dup}, and T_{ins} are the threshold values used in the proposed method for inter-frame video tampering detection. False negative rate (FNR) and false positive rate (FPR) are used for tuning the parameters. FNR indicates the proportion of wrong classification in positive cases and FPR indicates the proportion of wrong classification in negative cases. FNR and FPR are computed as:

$$FNR = \frac{FN}{TP + FN} \tag{3.14}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.15}$$

An ideal system should have low FNR and FPR.

T_{dup} is employed for distinguishing the duplicated frame sequences in the tampered video while T_{shuf} is for frame shuffling detection. We randomly chose 72 videos with various compression factors from frame duplicated videos. Likewise, 72 forged videos are randomly selected from frame shuffled videos. 30 videos from pristine category are also taken. FPR and FNR changes corresponding to various

values for T_{dup} and T_{shuf} are given in Fig. 3.14 and Fig. 3.15 respectively. In Fig. 3.14, FPR is rising after 0.95, and FNR obtained for this value is less. In Fig. 3.15, FPR is rising after 0.96, and FNR obtained for this value is fair.

T_{ins} is the threshold for identifying if the background visuals of inserted frames are similar to that of the pristine frames in the tampered video. 72 videos are taken from frame insertion tampered videos where 36 of them have similar background for inserted frames and the rest did not. In this context, TP (added frames having a different background scene as opposed to pristine are identified as such), TN (added frames having similar background as that of pristine are identified as such), and FP (added frames have similar background compared to pristine are classified as dissimilar) are taken accordingly. FPR and FNR estimated on various values for T_{ins} are given in Fig. 3.16. FNR is 0 for the threshold values chosen. However, FPR is high for large threshold values.

The threshold values for which we achieved almost equal error rate are chosen for performance evaluation of the proposed method. Therefore, 0.95, 0.96, and 0.88 are the values assigned to T_{shuf}, T_{dup}, and T_{ins} respectively for performance analysis.

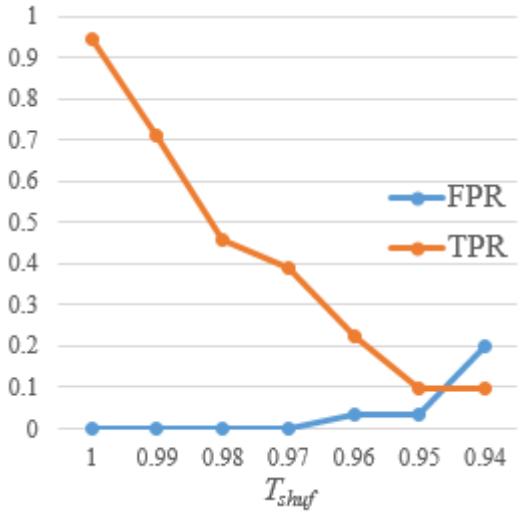


Figure 3.14: FNR and FPR variations on different threshold values for T_{shuf}.

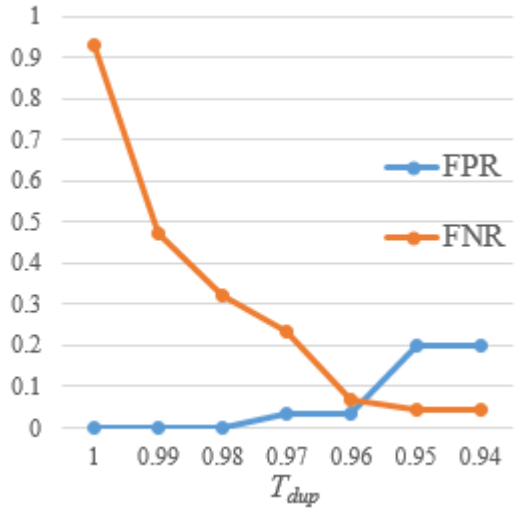


Figure 3.15: FNR and FPR variations on different threshold values for T_{dup}.

Figure 3.16: FNR and FPR variations on different threshold values for T_{ins}.

3.3.3 Performance Analysis and Comparison

Video sequences employed for tuning threshold values are not used in performance analysis. The summary of the dataset used for performance analysis providing the number of videos in each category is presented in Table 3.3.

Table 3.3: Summary of videos taken from our dataset on various inter-frame tampering and pristine video categories used for performance analysis

Pristine	**Forged Videos**				**Total**
Videos	**Insertion**	**Deletion**	**Duplication**	**Shuffling**	**Videos**
2346	6408	6480	6408	1944	23586

Table 3.4 shows the TPR obtained for frame insertion, deletion, duplication, and shuffling for CBR coded videos. B_1 and B_2 stands for bitrates in general for the first and second compression respectively. It can take values from $\{b_1, b_2, b_3\}$ as discussed in Subsection 3.3.1.2. The first value .921 is the TPR of videos with frame duplication which underwent MPEG-4 compression with b_1 bitrate for first compression and after editing they are saved in MPEG-4 format with bitrate b_1. Similarly, the second value .932 shows the TPR value of duplicated videos which underwent MPEG-4 compression with b_1 and b_2 bitrates in first and second compression respectively. MPEG-4-H.264 stands for videos which are compressed with MPEG-4 and H.264 in first and second compression respectively. H.264-MPEG-4 are the videos which underwent H.264 in its first compression and MPEG-4 in second respectively. H.264-H.264 represents videos which used H.264 encoder in its first and second compressions. Similarly, Table 3.5 shows the TPR obtained for frame insertion, deletion, duplication, and shuffling for VBR coded videos. C_1 and C_2 stands for Qscale (in case of MPEG-4) or crf (in case of H.264) values in general for the first and second compression respectively. It can take values from $\{c_1, c_2, c_3\}$ as discussed in Subsection 3.3.1.2. Table 3.4 and 3.5 show robustness of the proposed method to some degree of compression. The performance of proposed method is better when tampered videos are encoded with H.264 on second compression. The reason behind this is that H.264 provides better quality videos compared to MPEG-4. Also, the performance is slightly decreased in videos having $B_1 > B_2$ (CBR coded) or $C_1 < C_2$ (VBR coded). This is because of high compression in second encoding than first which affects tamper traces.

Table 3.4: TPR of proposed method on CBR coded videos

Forgery Type	**B_1/B_2**	**MPEG4-H.264**			**MPEG4-MPEG4**			**H.264-H.264**			**H.264-MPEG4**		
		b_1	b_2	b_3	b_1	b_2	b_3	b_1	b_2	b_3	b_1	b_2	b_3
Duplication	b_1	**.933**	**.955**	**.978**	.921	.932	.966	.921	**.955**	**.978**	.910	.933	.955
	b_2	**.910**	**.955**	**.978**	.899	.933	.955	.899	**.955**	.966	.888	.921	.944
	b_3	**.910**	**.933**	**.944**	.899	.910	.921	.899	.921	**.944**	.876	.899	.921
Deletion	b_1	**.955**	**.967**	**.978**	.933	.944	**.978**	.944	.956	**.978**	.933	.944	.956
	b_2	**.933**	**.944**	**.956**	.911	.933	.944	.911	**.944**	**.956**	.900	.922	.944
	b_3	**.922**	**.911**	**.922**	.889	.889	.900	.889	.900	**.922**	.878	.889	.900
Insertion	b_1	**.955**	**.978**	**.978**	.944	.955	.966	**.955**	**.978**	**.978**	.944	.955	.955
	b_2	**.933**	**.966**	**.978**	.910	.955	.966	**.933**	**.966**	.966	.910	.944	.955
	b_3	**.910**	**.910**	**.933**	.899	.899	.910	.899	**.910**	**.933**	.899	.899	.899
Shuffling	b_1	**.889**	**.926**	**.963**	.815	.852	.926	.852	.889	**.963**	.815	.852	.926
	b_2	**.852**	**.889**	**.926**	.815	.852	**.926**	.815	.852	**.926**	.778	.815	.889
	b_3	**.815**	**.852**	**.852**	.778	.815	.815	**.815**	**.852**	**.852**	.778	.815	.815

Table 3.6 provides the confusion matrix of the proposed method on pristine and various inter-frame tampered videos. It implies that our method is successful in inter-frame video tamper type classification. The 1^{st} row provides information on classification ability of the proposed method on pristine videos. It is the fraction of right classification (pristine) over the entire pristine videos given as input. The misclassified videos belong to the categories of frame insertion and deletion. Of the whole pristine videos, 0.9% and 1.7% are wrongly classified as frame insertion and deletion respectively. Likewise, the 2^{nd} row furnishes information on detection ability of the proposed scheme on duplicated videos. It is the fraction of right classification (frame duplication) over the total number of videos with frame duplication tampering. Here, the misclassified videos belong to pristine, insertion and deletion categories.

Table 3.5: TPR of proposed method on VBR coded videos

Forgery Type	**C_1/C_2**	**MPEG4-H.264**			**MPEG4-MPEG4**			**H.264-H.264**			**H.264-MPEG4**		
		c_1	c_2	c_3	c_1	c_2	c_3	c_1	c_2	c_3	c_1	c_2	c_3
Duplication	c_1	**.955**	**.921**	**.899**	.933	.910	.888	.944	.910	**.899**	.933	.910	.876
	c_2	**.989**	**.955**	**.933**	.955	.944	.899	.966	**.955**	.921	.921	.944	.899
	c_3	**.989**	**.966**	**.944**	.978	.955	.921	.978	.955	.933	.966	.944	.921
Deletion	c_1	**.967**	**.933**	**.911**	.956	.922	.889	**.967**	.911	.900	.944	.922	.889
	c_2	**.978**	**.944**	**.933**	.956	.933	.922	.967	.933	**.933**	.944	.933	.911
	c_3	**.978**	**.978**	**.933**	.967	.967	.922	**.978**	.967	.922	.967	.967	.922
Insertion	c_1	**.978**	**.933**	**.910**	.955	.944	.899	.966	.944	**.910**	.955	.944	.899
	c_2	**.989**	**.966**	**.955**	.978	.944	.933	.978	.955	.944	.966	.944	.933
	c_3	**1.00**	**.989**	**.955**	.989	.978	.944	.989	.978	.944	.978	.966	.944
Shuffling	c_1	**.926**	**.889**	**.815**	.889	.852	.778	**.926**	**.889**	**.815**	.852	.852	.778
	c_2	**.926**	**.889**	**.889**	.889	.889	.852	**.926**	**.889**	.852	.889	.852	.852
	c_3	**.963**	**.926**	**.889**	.926	**.926**	.852	**.963**	**.926**	.852	.926	.889	.852

Table 3.6: Confusion Matrix of the proposed method on our dataset in Table 3.3

Video Type	**Pristine**	**Insertion**	**Deletion**	**Duplication**	**Shuffling**
Pristine	**97.4**	0.9	1.7	–	–
Insertion	0.9	**94.7**	4.3	0.1	–
Deletion	3.4	3.1	**93.5**	–	–
Duplication	3.8	1.5	1.3	**93.4**	–
Shuffling	3.2	5.6	4.3	–	**86.9**

Table 3.7 provides information on inter-frame forgeries addressed by the methods in [12, 38, 102, 135, 138, 142, 144]. '✓' indicates tampering addressed and 'X' for those not addressed.

Table 3.7: Inter-frame video tampering addressed in existing methods and proposed method

Different Approaches	**Insertion**	**Deletion**	**Duplication**	**Shuffling**
Yu et al. [144]	X	✓	X	X
Yang et al. [142]	X	X	✓	X
Huang et al. [38]	X	✓	X	X
Chao et al. [12]	✓	✓	X	X
Wu et al. [138]	X	✓	✓	X
Wang et al. [135]	✓	✓	✓	X
Sitara & Mehtre [102]	✓	✓	✓	✓
Proposed	✓	✓	✓	✓

Table 3.8 provides performance comparison of the proposed method with those in [12, 38, 102, 135, 138, 142, 144]. The first column conveys information on the tamper types taken for performance evaluation when we compare our approach with existing methods. This is based on the kind of inter-frame tamper tackled by corresponding methods. As Wu et al. discussed frame deletion and duplication, we compare our method with their's in these two categories. Pristine videos are also considered along with the tampered videos in the respective groups (frame deletion and duplication) to conduct the performance evaluation. The performance comparison of our method with other existing methods are performed correspondingly. For evaluating precision, recall, F_1score, and accuracy, pristine videos are taken together with the inter-frame tamper types discussed in these methods. Table 3.8 implies that the proposed method surpasses existing methods.

The last row presents the performance of our method on all inter-frame tamper type under consideration. The existence of two abnormal peaks is compulsory for frame duplication detection in [138] and [135]. The frame duplicated videos in our dataset consists of some videos where the duplicated frames are placed either at the starting or ending of the video. Such videos have a single abnormal point and hence, the methods in [135, 138] classify it as frame deletion. The methods in [12, 102, 135, 138, 144] classified pristine videos consisting of zoomed frames as tampered. The methods in [12, 102, 135] classified the tamper type of those videos as frame insertion. Wu et al. [138] classified the tamper type as frame duplication. The methods for frame insertion detection discussed in [12, 144] failed in cases where static frames having similar background scene as that of the pristine frames in the tampered video are inserted. [38, 142] are sensitive to compression. The method for frame shuffling detection in [102] uses direct frame correlation. Hence, it is sensitive to compression.

The threshold values can also be taken dynamically depending on the compression ratio of a video under investigation. If this video is of good quality with less compression, then we can chose tight thresholds. If not, a loose thresholding can be applied.

If we apply inter-frame video tampering at multiple temporal locations, it will contribute several (≥ 2) abnormal peaks in its VFIs. The grouping of these irregular peaks is the major concern to be rectified first. Consider a case in which a perpetrator deletes frame sub-sequences from two distinct temporal positions. In this case, two tamper points are introduced in VFIs as well as VPA. Hence, our method classifies such a video as frame insertion. Let us consider another case, application of frame duplication at two distinct temporal positions. If all of the four t_p are detected, then our method can identify the two candidate clips used for tampering. If one or more of t_p are missed, grouping and testing of frames are to be modified correspondingly.

The proposed scheme for zooming detection rightly classified all videos (22) with zooming as such. Moreover, it did not return any false positives on normal videos. Therefore, the proposed system is apt for zooming detection.

Video editing tools such as Blender, Adobe Premiere Pro, etc. can be used to create zoomed frames in a video. A perpetrator may create an inter-frame

Table 3.8: Performance comparison of the proposed method with existing methods based on the type of inter-frame video tampering addressed

Forgery Type	Different Approaches	Recall	Precision	Accuracy	F_1score
insertion	Huang et al. [38]	.913	.964	.911	.938
	Proposed	**.947**	**.99**	**.954**	**.968**
deletion	Yu et al. [144]	.913	.909	.868	.911
	Proposed	**.935**	**.99**	**.945**	**.962**
duplication	Yang et al. [142]	.846	.965	.865	.902
	Proposed	**.934**	**.989**	**.945**	**.961**
deletion & insertion	Chao et al. [12]	.874	.941	.847	.907
	Proposed	**.941**	**.995**	**.946**	**.967**
duplication & deletion	Wu et al. [138]	.896	.949	.871	.921
	Proposed	**.934**	**.995**	**.941**	**.964**
duplication, deletion & insertion	Wang et al. [135]	.889	.963	.871	.925
	Proposed	**.939**	**.997**	**.942**	**.967**
insertion, deletion, duplication, & shuffling	Sitara & Mehtre [102]	.904	.987	.903	.944
	Proposed	**.932**	**.997**	**.936**	**.964**

tampered video by making use of these artificially zoomed frames. He/she can use these frames for inter-frame tampering such as duplication, insertion, or shuffling. Irrespective of the nature of origin (natural or artificial), existing methods classify videos with zooming as tampered. As zooming detection is incorporated to inter-frame tampering detection, our method classifies such videos as genuine. Hence,

inter-frame tampering using artificially zoomed frames serves as an anti-forensic approach to the proposed method.

3.4 Summary

Frame shuffling detection that was an unresolved challenge in video tampering detection is discussed in this chapter. Videos containing frames captured during camera zooming added false positives in previous works. Sudden zooming in such videos caused abnormalities in the features used for video tampering detection. Hence, such pristine videos get incorrectly classified as tampered. We fixed this issue by introducing a method for zooming detection and consolidated the same with video tampering detection. A dataset consisting of 23586 videos that comprise pristine and tampered videos with various types of inter-frame alterations are used for testing the proposed method.

Experimental results show that our method performs better compared to the existing techniques. Our method is capable of handling videos having AGOP structure. In a tampered video, it can distinguish the type of inter-frame tampering. We are successful in identifying the temporal locations of tampering. In frame shuffling and duplication, the original frames utilized for performing these forgeries are also discovered. We have tested the robustness towards VBR, and CBR coding schemes and obtained satisfactory results.

In future, we intend to explore the feasibility of detecting multiple forgeries in single video using our methods. The detection of frame shuffling forgery having single tamper point is another research path. The proposal of tamper detection methods to improve the robustness towards the usage of synthetic zoomed frames in inter-frame forgeries.

Chapter 4

Differentiating Synthetic Zooming from Natural Camera Zooming for Video Tampering Detection

Apart from the inter-frame forgeries like frame deletion, duplication, insertion and shuffling, we have identified that inter-frame forgeries can also be performed using upscale-cropped frames. If the scale factor (SF) used for upscale (resampling) forgery is varying, then it resembles natural camera zooming (concept discussed in Section 1.2 of Chapter 1). Video editing tools like Adobe Premiere Pro, Blender etc can be used for creating tampered videos with synthetic zooming. This kind of tampering creates forged frames that merge seamlessly with the pristine frames in video without introducing any visual distortion in the tamper video. Sample frames taken from synthetically zoomed video created using Adobe After Effects are given in Fig. 1.4 and 1.5 of Chapter 1. In case of tampered videos with synthetic zooming, the investigator may discard the results of passive video tampering detection methods [1, 20, 50, 109], in the belief that they are genuine frames due to camera zooming, leading to false negatives. A method for detecting zoomed frames in videos under investigation is discussed in [102] for reducing false positives due to genuine camera zooming. This contributed false negatives when those zoomed frames are created synthetically. In this chapter, we propose

a passive method for differentiating synthetic and natural zooming which can be incorporated with inter-frame video tampering detection techniques for reducing the false negatives caused due to synthetic zoomed frames.

The proposed method for differentiating synthetic zooming from optical camera zoom identifies the location (frame numbers) of zoom-in and zoom-out frames in a video. Applying resampling detection method on every frame in a video requires high computation and also is a time consuming process. As the duration of video increases, so is the number of frames in it. Hence, finding the location of frames which need to be examined for resampling may reduce time complexity. The blocking artifact that arises due to compression is the major challenge faced by most resampling detectors [29, 51, 52, 53, 71, 80]. It introduces periodic peaks in frequency domain. Earlier methods ignored such peaks. Hence, when these peaks overlap with those of tamper (resampling) artifacts, the candidate image will be misclassified as pristine. The goal is to design and develop a simple and effective resampling detector which is capable of revealing resampling artifacts even in the presence of compression. To identify the location of zoom-in and zoom-out frames in video, the method discussed in Subsection 3.2.4 of Chapter 3 is used. The frames in between zoom-in and zoom-out operations are taken for further processing. The presence of pixel variance correlation and SPN variations in these frames are analyzed for determining forgery. The frames recorded during zoom-in and zoom-out lens adjustments are not used. The reason for this is explained with an example in Subsection 4.1.1. The proposed method is insensitive to kind or quality of compression, visual contents and characteristics of the videos.

The rest of this chapter is organized as follows. Section 4.1 discusses the proposed methods for differentiating synthetic and natural zooming. Section 4.2 deals with experimental results and discussion on performance of the proposed method. Conclusions and future research recommendations are summarized in Section 4.3.

4.1 Proposed method for Synthetic Zooming Detection

Interpolation algorithms estimate the pixel values at intermediate positions from its neighborhood. This process introduces dependencies between adjacent pixels. The strength of dependence varies periodically. The resampling detection methods that rely on periodic peaks will suffer from blocking artifacts due to compression. Hence, the proposed method is designed such that it will not depend on the periodicity of peaks. The flowchart of proposed method is given in Fig. 4.1. Given a video, it is first analyzed using the zooming detection method discussed in Section 3.2.4 of Chapter 3 for identifying whether the video contains frames with zoom-in and zoom-out operations. If such frames exist, then some random frames which appear in between the end of zoom-in and the beginning of zoom-out operations are taken for further processing. These frames are indicated as candidate frames in Fig. 4.2 which is a pictorial representation of a video sequence with N frames. $t1$ indicates the last frame of zoom-in operation. $t2$ denotes the first frame of zoom-out operation. The proposed method will then check for pixel variance correlation and SPN variations on random frames between $t1$ and $t2$. If both of these exist, then those frames are classified as candidates of synthetic zooming.

4.1.1 Pixel Variance Correlation Detector

The existence of periodicities in the variances of n^{th} order derivatives of resampled images are studied and proved in [29, 70, 71]. The row-wise and column-wise second order derivatives (D_{xx} and D_{yy}) of a frame, I of size $r \times c$, are obtained. The horizontal derivative kernel, h_{xx} is $[1, -2, 1]$ and vertical h_{yy} is h_{xx}^T where T denotes transpose. The mean of magnitudes (m_x of size $1 \times r$) of D_{xx} are obtained row-wise. The autocorrelation sequences along each row of D_{xx} are computed using

$$ac_x(i,k) = \frac{\sum_{j=1}^{n-k}[(D_{xx}(i,j) - m_x(i)) \cdot (D_{xx}(i,j+k) - m_x(i))]}{\sum_{j=1}^{n}(D_{xx}(i,j) - m_x(i))^2} \tag{4.1}$$

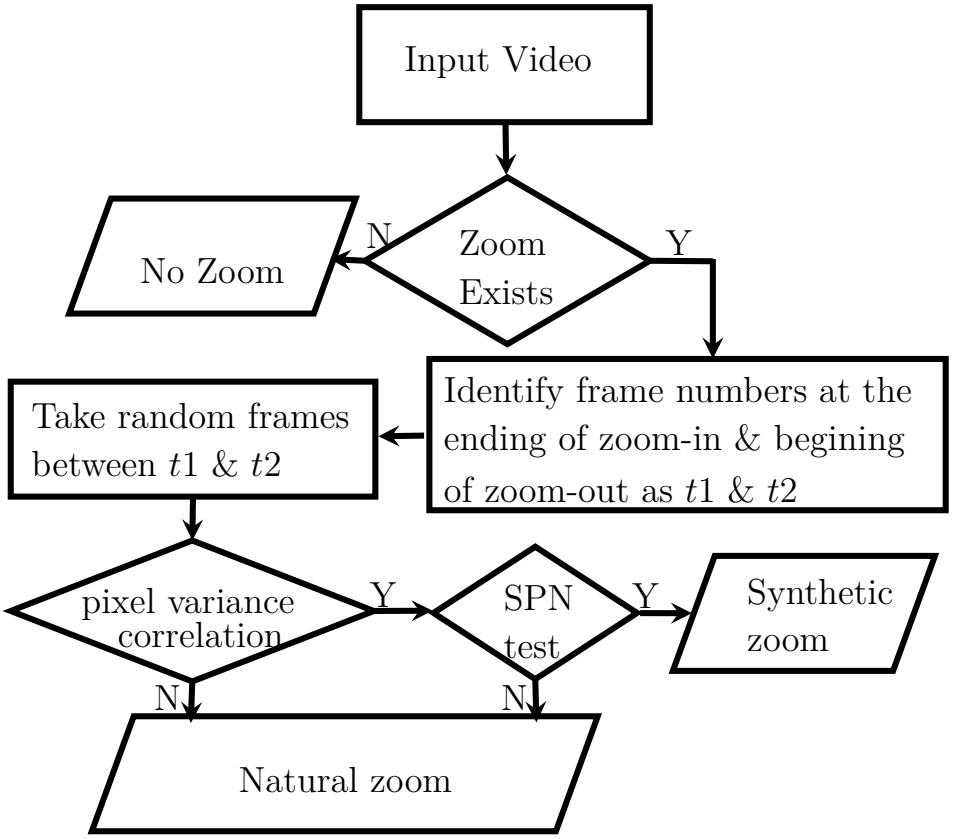


Figure 4.1: Flowchart of the proposed synthetic zooming detection method

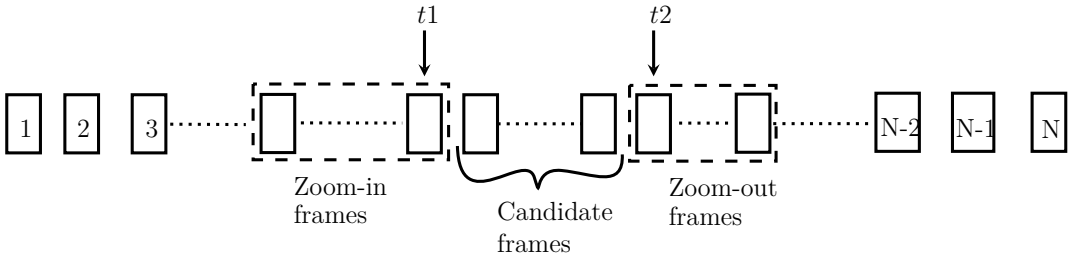


Figure 4.2: Candidate frames for synthetic zooming detection

where $ac_x(i,k)$ is the autocorrelation value of i^{th} row in D_{xx} with lag k. Similarly, $ac_y(j,k)$ is computed from the j^{th} column of D_{yy}. After obtaining the autocorrelation values of all rows and columns of D_{xx} and D_{yy}, their FFT coefficients (F_x and F_y) are computed. The row-wise mean of F_x is obtained to get a $1-D$

vector, mF_x of length r. Similarly, the column-wise mean of F_y is taken (mF_y). For thresholding, hmF_x and hmF_y are obtained by dividing mF_x and mF_y with the highest value in them. The counts of all values in hmF_x and hmF_y above a threshold T_1 are taken separately. If 96% of these values are above T_1, then it is classified as a natural frame. If not, continue the synthetic zoom test for SPN variation. The procedure for pixel variance correlation detection is given in Algorithm 8.

Algorithm 8 Procedure for pixel variance correlation detection

1: Compute horizontal and vertical second derivatives D_{xx} and D_{yy} of frame I using h_{xx} and h_{yy} respectively;
2: Compute the autocorrelation sequences of D_{xx} and D_{yy} in row-wise and column-wise respectively;
3: Compute FFTs of ac_x and ac_y and obtain their row-wise and column-wise mean values to mF_x and mF_y respectively;
4: $hmF_x \leftarrow mF_x/max(mF_x)$; $hmF_y \leftarrow mF_y/max(mF_y)$;
5: Assign the count of values in hmF_x and hmF_y which are greater than T_1 to cx and cy respectively;
6: **if** $(cx/length(mF_x)) > .96$ and $(cy/length(mF_y)) > .96$ **then**
7: Natural frame;
8: **else** Proceed with SPN test;
9: **end if**

Figure 4.3 and 4.4 show the distributions of mean FFT values of pixel variance correlation in normal, natural camera zoom and synthetic zoomed frames. The horizontal axes in 2^{nd} and 3^{rd} row figures denote the length of hmF_x and hmF_y respectively. The vertical axes hold their normalized magnitudes. From Fig. 4.3, it is evident that normal and natural zoomed frames share similar pattern. In Fig. 4.4, the existence of high pixel correlation in synthetic zoomed frame is introducing a single high peak in its FFT. Automatic lens adjustments during zoom-in and zoom-out operations is also contributing similar single peaks in the FFTs of pixel variance correlation. This is because of lens defocusing during lens adjustments. It introduces blur in the recorded frames which in-turn results in high pixel correlation. An example of such a frame with its corresponding hmF_x

and hmF_y are given in Fig. 4.4 (d), (e) and (f) respectively. This is the reason that frames recorded during zoom-in and zoom-out operations are not used for synthetic zooming detection in our work.

4.1.2 SPN test

The imaging sensor (CCD or CMOS) in digital camera contains picture elements (pixels) for collecting photons. These are made of silicon wafers. Due to its inhomogeneity, they exhibit different sensitives to light which generates pattern noise called sensor pattern noise (SPN). Every frame recorded by the camera holds SPN and it varies from camera to camera. Usually, it is used as a forensic feature for source camera identification (SCI). For a frame, I, SPN can be calculated as

$$SPN_I = \frac{W\hat{I}}{(\hat{I})^2} \text{ where } \hat{I} = F(I), W = I - \hat{I} \tag{4.2}$$

$F()$ is the wavelet-denoising filter [18]. For problems like SCI, the average of SPNs extracted from a sequence of frames forms SPN of the video clip. It requires precise SPN estimation for successful device linking. Here, the interest is on finding frames with abnormal noise patterns. Hence, precise SPN is not required. The procedure for finding abnormality in SPN is given in Algorithm 9.

Algorithm 9 Procedure for detecting abnormality in SPN

1: Compute SPN of frame I (SPN_I) using Eqn. 4.2;
2: Compute horizontal second derivative, S_{xx}, of SPN_I using h_{xx};
3: Obtain row-wise autocorrelation sequences of S_{xx} to ac_s;
4: Compute FFT of ac_s and obtain row-wise mean to mF_s;
5: $hmF_s \leftarrow mF_s/max(mF_s)$;
6: Assign the count of values in hmF_s which are greater than T_2 to cs;
7: **if** $(cs/length(mF_s)) > .96$ **then**
8: Synthetic upscale-cropped frame;
9: **else** Natural frame;
10: **end if**

The distributions of mean FFT co-efficients of SPN autocorrelation sequences (hmF_s) in normal, natural camera zoom and synthetic zoomed frames are shown

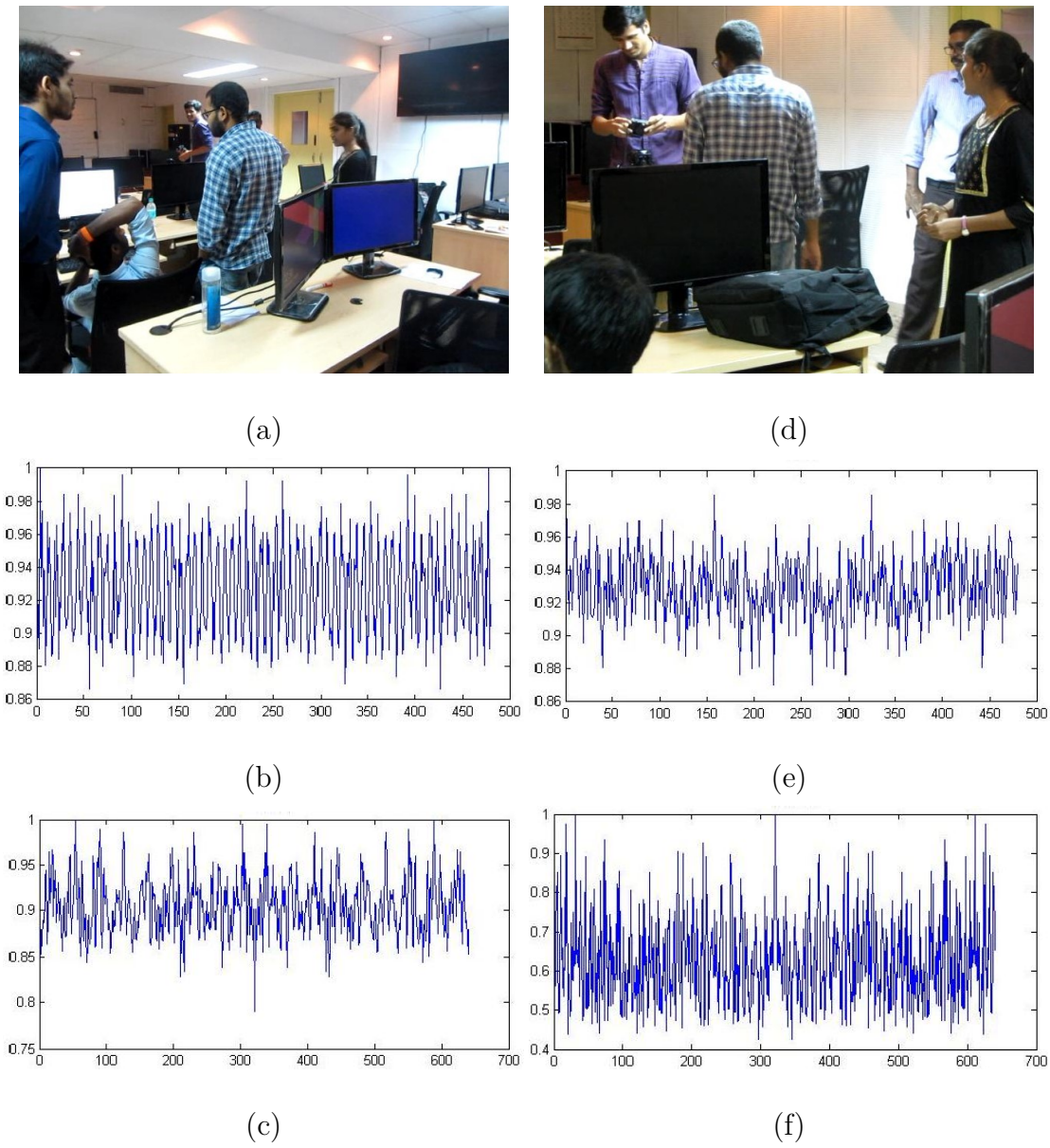


Figure 4.3: Pixel variance correlation in normal frame and natural camera zoomed frame. (a) Normal frame, its hmF_x and hmF_y are given in (b) and (c) respectively; (d) Natural optical camera zoom frame, its hmF_x and hmF_y are given in (e) and (f) respectively.

in Fig. 4.5. The horizontal axis denotes the length of hmF_s which is r of I. The vertical axes hold their normalized magnitudes. From Fig. 4.5 (a) and

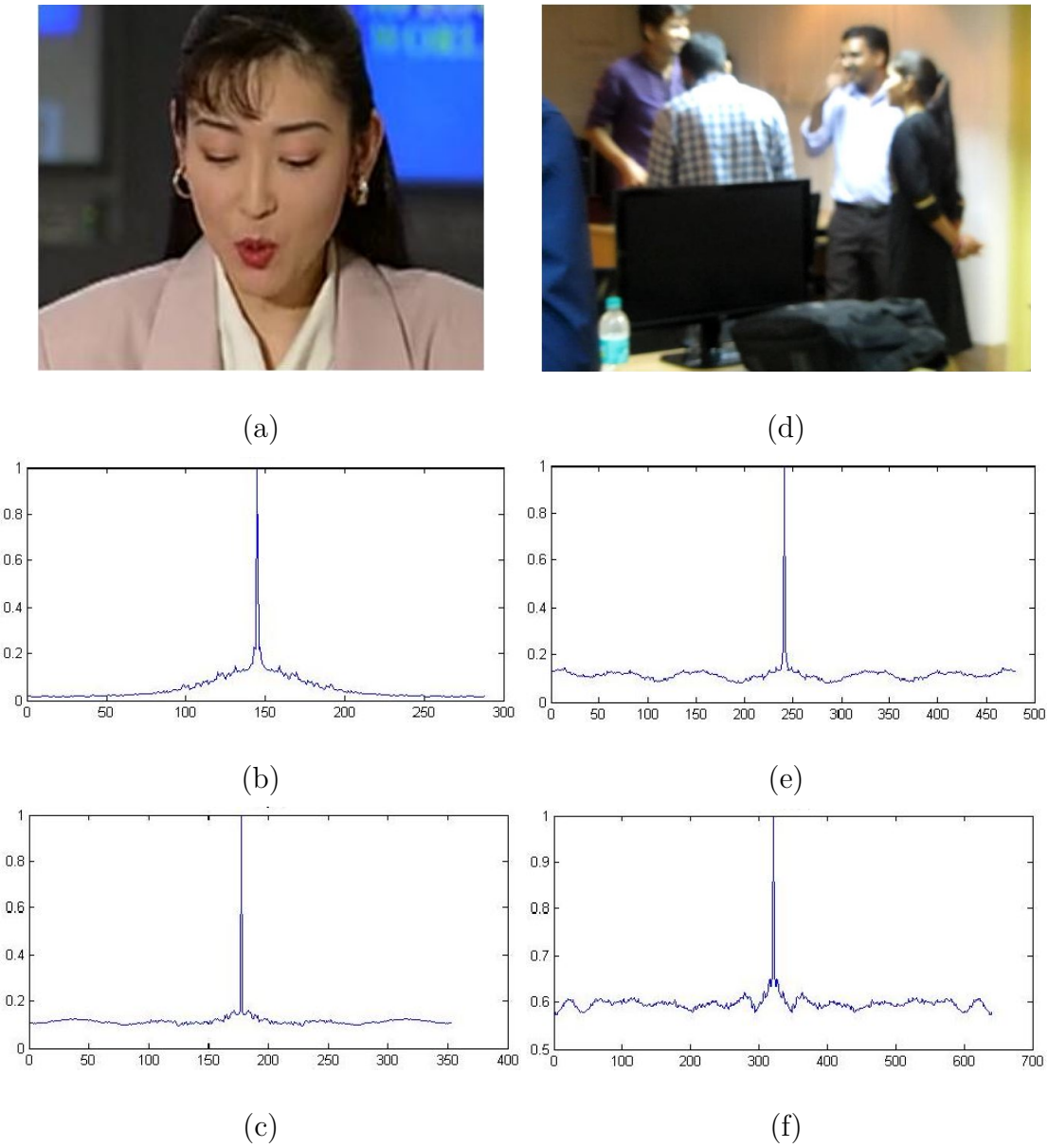


Figure 4.4: Pixel variance correlation in synthetic zoomed frame and frame recorded during natural camera zoom-in activity. (a) Synthetic zoomed frame, its hmF_x and hmF_y are given in (b) and (c) respectively; (d) Natural camera zoom-in with len defocus and its corresponding hmF_x and hmF_y are given in (e) and (f) respectively.

(b), it is evident that normal and natural zoomed frames share similar pattern. There exists a single peak in their SPN distributions. The resampling process is affecting the SPN distribution in synthetic zoomed frames. hmF_s obtained from the synthetic zoomed frame in Fig. 4.4(a) is shown in Fig. 4.5(c). SPN distribution in frames recorded during zoom-in and zoom-out operations also share similar pattern as that of the natural frames. hmF_s of such a frame in Fig. 4.4(d) is given in Fig. 4.5(d). It shows that lens defocusing during auto lens adjustments has no impact on SPN.

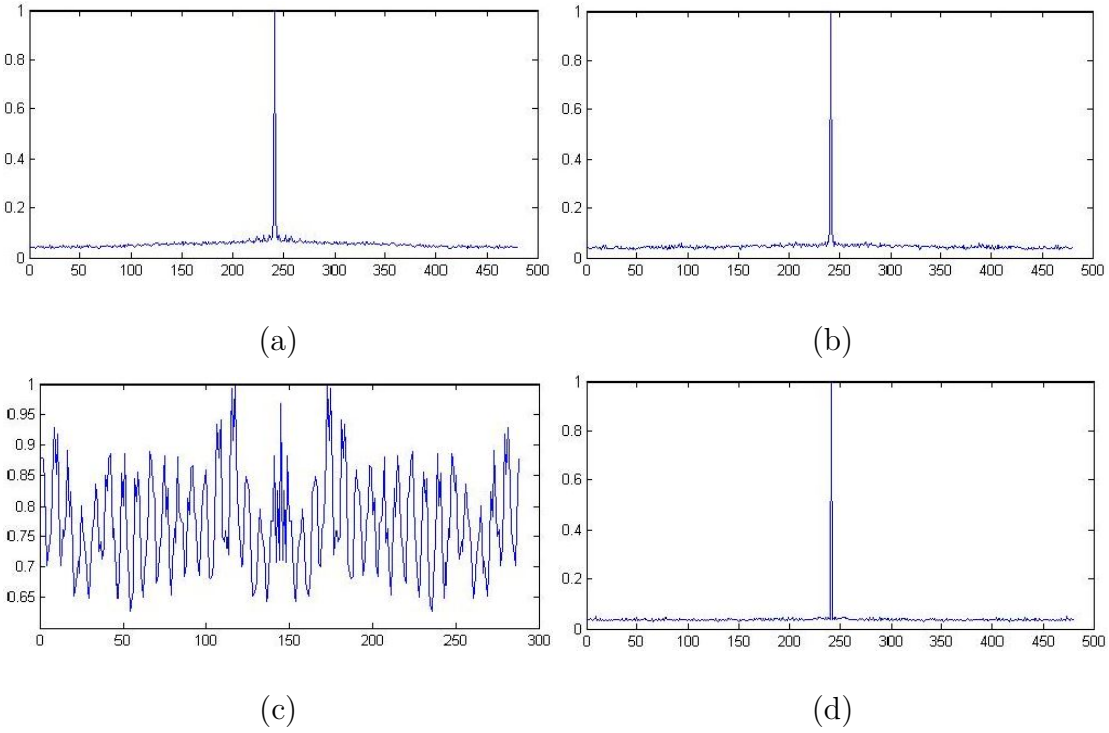


Figure 4.5: SPN variation in normal, natural camera zoom and synthetic zoomed frames. (a) hmF_s of normal frame in Fig. 4.3(a); (b) hmF_s of natural camera zoom frame in Fig. 4.3(d); (c) hmF_s of synthetic zoomed frame in Fig. 4.4(a); (d) hmF_s of natural camera zoom-in with len defocus in Fig. 4.4(d).

The count of all values in hmF_s that are above a threshold T_2 is taken for objective detection of tampered (upscale-cropped) frames. If 96% of these values are greater than T_2, then such a frame is classified as a synthetic resampled frame. If not, it is classified as a natural frame.

4.2 Experimental Results and Discussion

The proposed method is implemented using MATLAB R2013a (8.1.0.604), performed on 3.40 GHz Intel Core i7 PC. For parameter tuning and performance evaluation of the proposed method, the inter-frame video tamper detection dataset discussed in Subsection 4.2.1 of Chapter 2 is used. All video frames are converted to gray scale before processing it for zooming detection. Here, the description of TP, FN, TN and FP are as follows when synthetic zooming is considered as the positive class:

- **TP** - A synthetic zoomed video given to the proposed method for synthetic zooming detection classified as synthetic.
- **TN** - A video with optical zooming classified as natural.
- **FP** - A video with optical zooming classified as synthetic.
- **FN** - A synthetic zoomed video classified as natural.

4.2.1 Video Dataset Created

Public datasets are not available for synthetic zooming detection. Hence, we have created our own dataset. Sony Cybershot DSC-WX500 and Canon PowerShot SX230 HS cameras were used for creating this dataset. The videos are recorded at a resolution of 1920×1080 and 640×480 using the former and latter respectively. We recorded 40 video sequences with natural camera zooming and 20 without zooming. The video sequences without zooming are used for creating synthetic zoomed videos by using Adobe After Effects tool.

To increase the videos in dataset, we have downloaded 20 pristine videos (352×288 resolution) without zooming from [118, 145]. These downloaded videos are used for creating synthetic zoomed videos using Adobe After Effects tool. Bridge-Close, Akiyo, Highway, Coastguard, Silent, Carphone, Galleon, Mother and Daughter, Claire, Container, Sign-Irene, News, Foreman, Pamphlet, Grandma, Hall, Husky, Paris, Bowing, and Bridge-Far are the videos downloaded from [118, 145]. These videos are in YUV uncompressed format with

CIF (352×288) resolution. The resolution of videos from PETS2001, PETS2007 and PETS2009 are 768×576, 720×576 and 768×576 respectively.

All pristine and synthetic videos with zooming are compressed using MPEG-4 and H.264 formats with CBR and VBR coding. To study the system performance with noise, we added Guassian noise ($\sigma^2 = 0.02, 0.03, 0.04$), Speckle noise ($\sigma^2 = 0.04, 0.05, 0.06$) and salt & pepper noise with varying densities ($d = 0.03, 0.04, 0.05$) to the original and resampled videos with zooming. From a video sequence, we created 9 noisy variants of it. With the application of compression and noise settings on pristine and tampered videos, a total of 1600 natural and 1600 synthetic zoomed videos are created. Table 4.1 gives details of videos used for testing.

Table 4.1: Details of video dataset for synthetic zooming detection

Video Zoom Type	**Video Source**	**Number of Videos**	**Number of Videos with Noise & Compression Variants**
Natural camera zoom	recorded by us	40	1600
No Zoom	recorded by us	20	–
	downloaded from [118, 145]	20	–
Synthetic zoom	created by us from downloaded and recorded videos without zooming	40	1600

4.2.2 System Behavior on Noise Addition

The variations in FFT plots of pixel variance correlation on different noise types considered in Subsection 4.2.1 of Chapter 2 are shown in Fig. 4.6, 4.7 and 4.8. Figure 4.6 (a), (b) and (c) show that noise has no impact on the pixel correlation plot of synthetic zoomed frames. Noise addition affected pixel variance correlation of normal (Fig. 4.7 (a), (b) and (c)) and natural camera zoom frames (Fig. 4.8

(a), (b) and (c)). It resembles to that of the pixel variance correlation in synthetic zoomed frames. The denoising method proposed in [18] is applied on frames so that the pixel variance correlation test when applied on denoised frames may give the intended results (plots) of pixel correlation for natural frames without affecting that of the synthetic frames. Figure 4.7 (d), (e) and (f) are pixel variance correlations of the corresponding denoised versions of normal frames. And Fig. 4.8 (d), (e) and (f) are pixel variance correlations of the corresponding denoised versions of natural camera zoomed frames. From the figures, it is clear that denoising resolves the issue. Also, Fig. 4.6 (d), (e) and (f) show that denoising have not affected the plots of resampled frames.

4.2.3 System Behavior with different Interpolation methods

An original frame taken from Coastguard video [118, 145] is upscaled with SF 1.4 using different interpolation methods like nearest-neighbor, bilinear and bicubic. The resampled frames obtained are cropped to the size of its original frame. The variations in FFT plots of pixel variance correlation and SPN with these upscale cropped frames are shown in Fig. 4.9. It shows the effectiveness of proposed method on different interpolation methods.

4.2.4 Performance Evaluation

One can make subjective evaluation based on the plots of pixel variance correlation and SPN distributions of a frame. For objective evaluation of the proposed method, 40 videos each are taken from the natural and synthetic zoom categories. False Positive Rate (FPR) and False Negative Rate (FNR) obtained over these videos on various threshold values for T_1 and T_2 are given in Fig. 4.10 and Fig. 4.11 respectively. FNR and FPR are computed using Eq. 3.14 and Eq. 3.15 respectively (discussed in Sub-section 3.3.2 of Chapter 3). The threshold values for which we got equal error rate are taken for performance evaluation of the proposed system on the entire dataset. Hence, T_1 and T_2 are set to 0.3 and 0.2 respectively. This is the first work on synthetic zooming detection in videos and

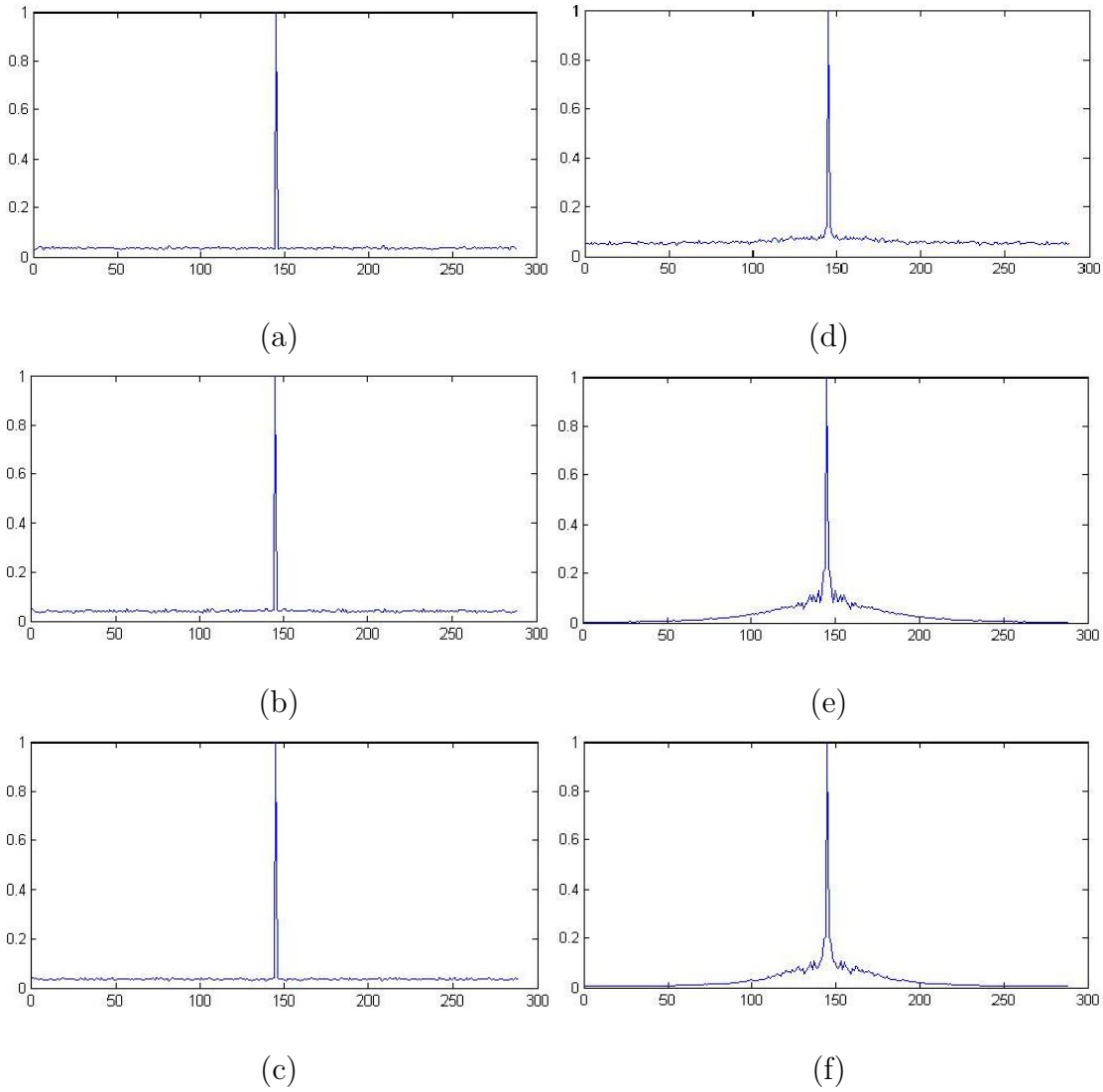


Figure 4.6: FFT plots of Pixel variance correlation in synthetic zoomed frames with different noise types. (a), (b) and (c) are pixel correlation distributions of synthetic zoomed frames corrupted with gaussian ($\sigma^2 = 0.04$), speckle ($\sigma^2 = 0.05$) and salt & pepper ($d = 0.05$) noises respectively. (d), (e) and (f) are that of its corresponding denoised frames.

hence a comparative study cannot be done. The performance of the proposed method on the dataset discussed in Subsection 4.2.1 is given in Table 4.2. The performance is evaluated by computing precision (P), recall (R) and F_1 score

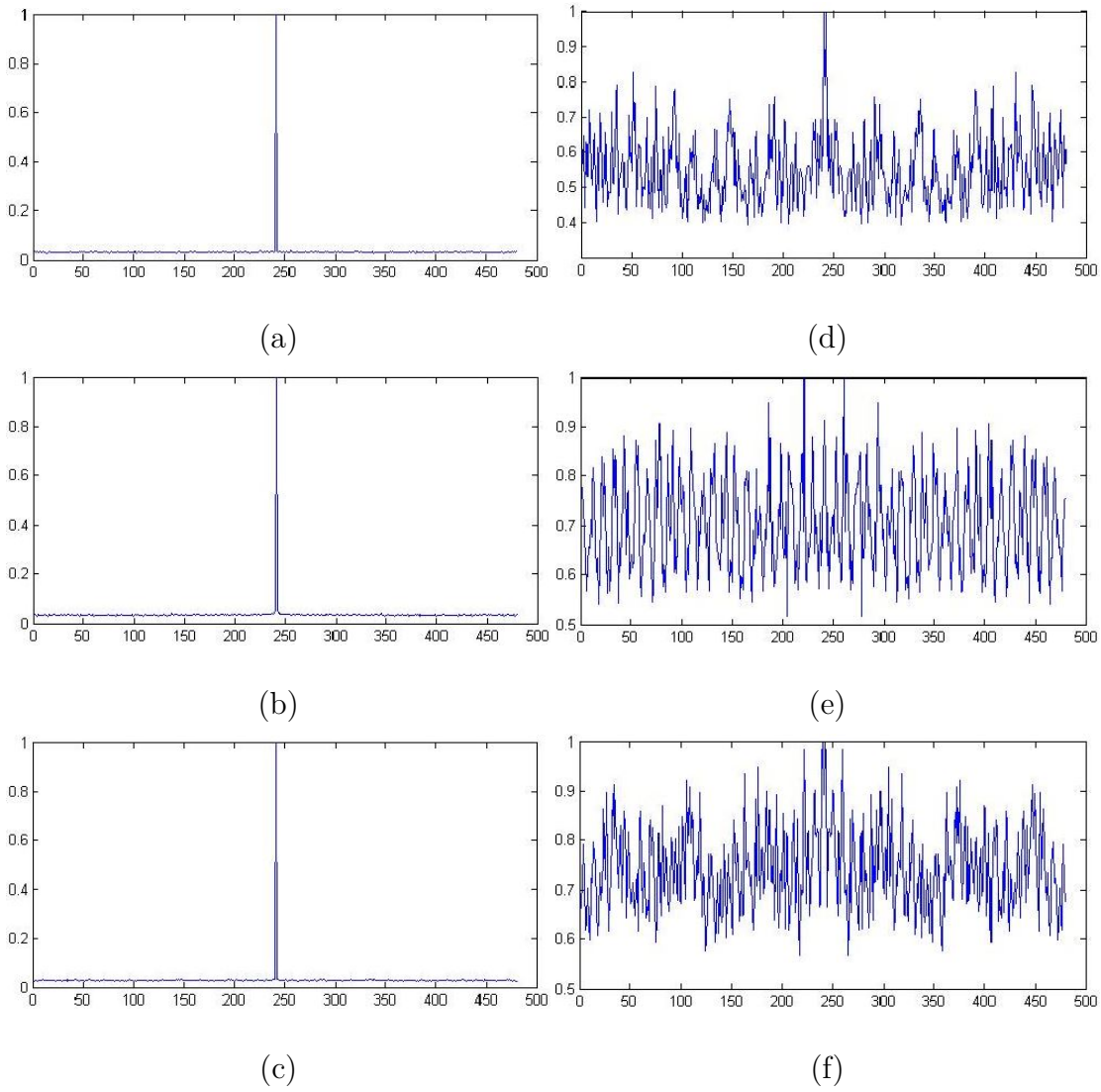


Figure 4.7: FFT plots of Pixel variance correlation in normal frames with different noise types. (a), (b) and (c) are pixel correlation distributions of normal frame corrupted with gaussian ($\sigma^2 = 0.02$), speckle ($\sigma^2 = 0.04$) and salt & pepper ($d = 0.05$) noises respectively. (d), (e) and (f) are that of its corresponding denoised frames.

(F_1). P, R, and F_1 score are computed using Eq. 2.2, Eq. 2.1 and Eq. 2.4 respectively (Sub-section 2.8 of Chapter 2). The first and second rows show the result analysis on CBR and VBR coded videos in the dataset respectively. The last row

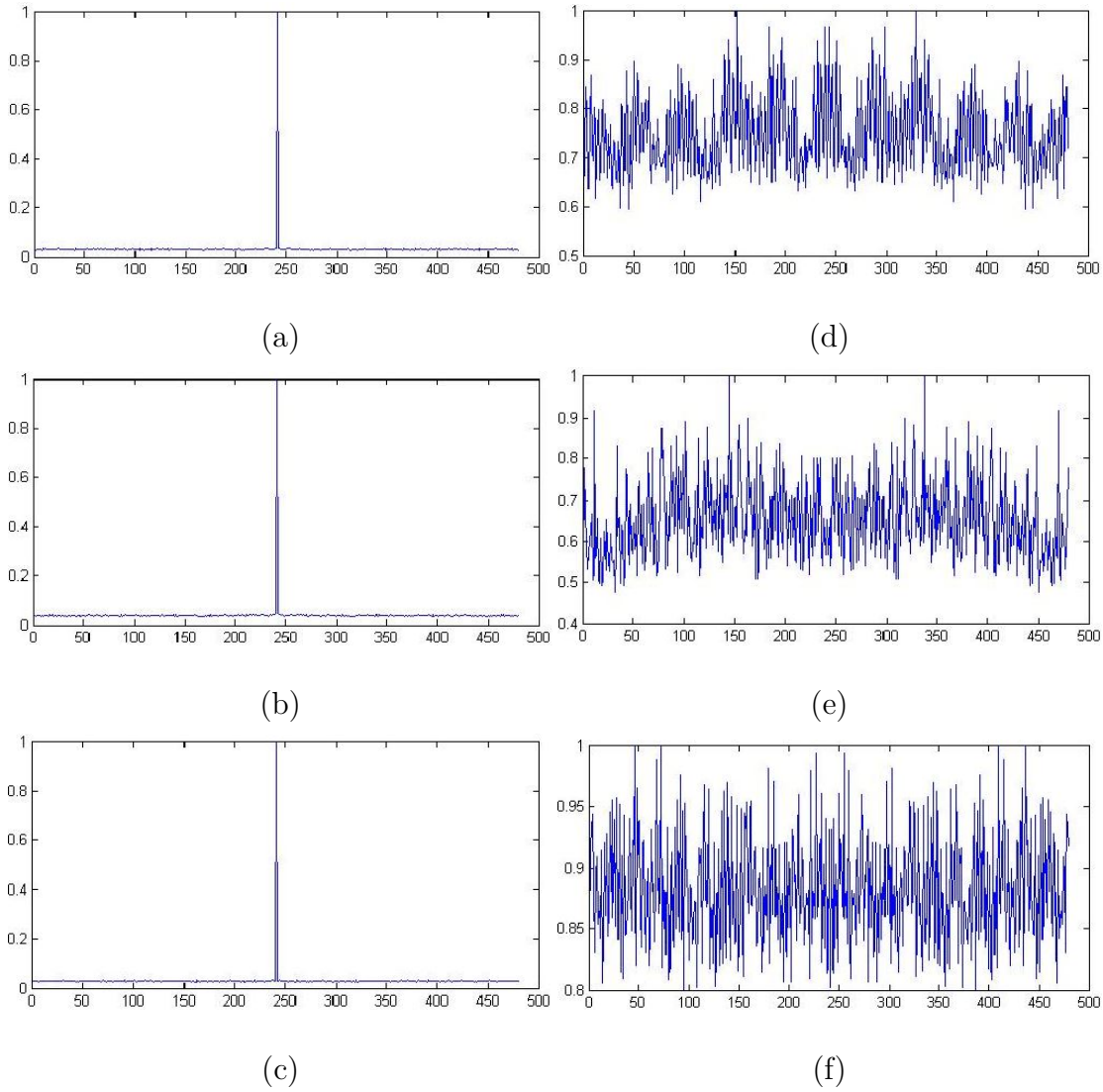


Figure 4.8: FFT plots of Pixel variance correlation in natural camera zoomed frames with different noise types. (a), (b) and (c) are the pixel correlation distributions of natural camera zoom frame corrupted with gaussian ($\sigma^2 = 0.02$), speckle ($\sigma^2 = 0.04$) and salt & pepper ($d = 0.05$) noises respectively. (d), (e) and (f) are that of its corresponding denoised frames.

is that obtained over the entire videos in the dataset.

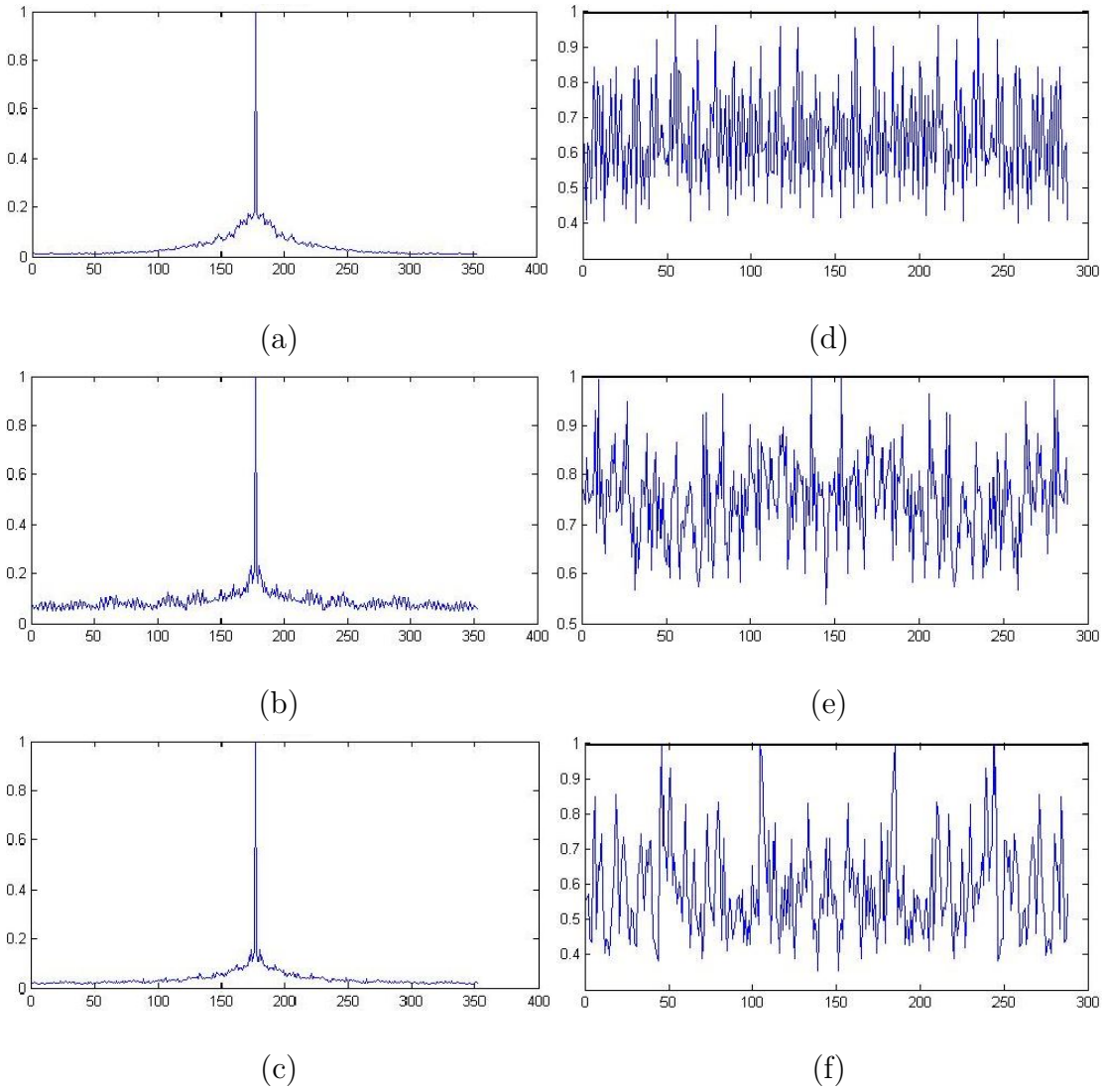


Figure 4.9: FFT plots of Pixel variance correlation and SPN using different interpolation methods with SF = 1.4. (a), (b) and (c) are the pixel correlation distributions of nearest neighbour, bilinear and bicubic interpolations respectively. (d), (e) and (f) are their corresponding SPN distribution.

4.3 Summary

Synthetic zooming detection, hitherto an unexplored area in video forgery detection is discussed. A forged video created by perpetrator utilizing synthetic zooming is very much appealing as a genuine video. The video forgery detection

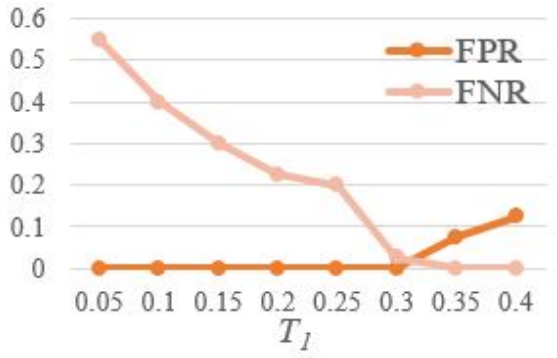


Figure 4.10: FPR and FNR variations on different threshold values for T_1

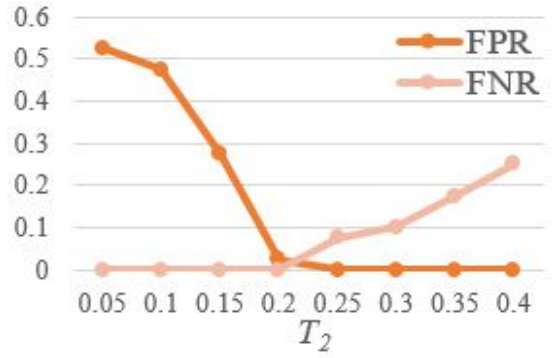


Figure 4.11: FPR and FNR variations on different threshold values for T_2

Table 4.2: Performance evaluation of the proposed method

Bitrate	**TP**	**FN**	**TN**	**FP**	***P***	***R***	$\boldsymbol{F_1}$
Constant	759	41	747	53	.935	.949	.942
Variable	764	36	762	38	.953	.955	.954
Total	1523	77	1509	91	.944	.952	.948

algorithms may fail in providing the exact classification of such a video as forged. Hence, the application of this concept as anti-forensics on video forgery detection fosters the relevance of differentiating synthetic zooming from natural zooming. The proposed system is tested on a dataset consisting of 3200 videos. Experimental results show the effectiveness of proposed system on various noise types

and compression settings.

Exploration of anti-forensic methods that can be applied over the proposed system is considered as future work. The methods for countering the anti-forensic aspects thus found is also under consideration.

Chapter 5

Camera Tampering Detection in Static Video Surveillance Systems

Video surveillance systems are installed in public and private places for ensuring safety and security or monitoring the areas of interest. These recordings are often produced as evidence in the court of law in case of crimes.In most cases, manually monitoring a video surveillance system is a tiresome and unfeasible task. The operator concerned may have to monitor the videos recorded by several cameras simultaneously. The visuals recorded by these cameras may be of the same FOV in various viewing angles or different FOVs. Studies in [113] indicate that the efficacy of manual monitoring is very low. Surveillance camera sabotage/tampering is a deliberate action on a surveillance camera to change the proper working of camera thereby altering the images recorded and processed by the surveillance systems. These actions include: camera moved/displaced, partial or full camera occlusion and camera out-of-focus/defocus and so on as discussed in Section 1.3 of Chapter 1. Camera displacement or occlusion will produce newer frames which are partially or totally distinct from the intended FOV. Some of these may occur accidentally, but detecting them is equally important as the video quality and scene/area under surveillance have to be properly recorded.

Automatic analysis and detection of camera tampering in real-time is very important. It helps in preventing a crime and alert operator so that he/she

may not miss any suspicious events. Apart from detection accuracy, the time complexity and false alarm rate of camera tamper detection algorithms are other factors to be taken care of. In most cases, this method has to be implemented as part of a surveillance system which has multiple cameras. Since, real-time detection of suspicious activities is required in such systems, the time complexity of the proposed method should be low. Coming to false alarm rate, if it is high the operator may loose interest in the automated system and he/she may ignore the future warnings from the system. Reducing false alarm rate is a challenge to the automated system as false alarms can be easily triggered due to illumination changes, crowd, large objects passing through the scene etc.

Figure 5.1 shows the block diagram of camera tampering detection technique enabled video surveillance system. Several other video surveillance algorithms for safety and security [3, 16, 28, 45, 57, 58, 62, 84, 90, 147] may also process the frames captured by surveillance camera. If a perpetrator tampers the surveillance camera, then the frames recorded during these camera tamper events may or may not be useful for video analysis and forensic investigation. Hence, we place camera tamper detection functionality as an add-on to the currently existing functionalities of a video surveillance system. This if implemented can trigger an alarm for camera tamper events in real-time without affecting the normal operations. It serves as a crime prevention mechanism which enables timely operator intervention. Despite the detection and activation of the alarm associated with camera tamper anomaly, it can also process the incoming frames with other algorithms for safety and security. This way, it may be possible to collect additional information about an event. We can collect the visuals recorded during camera tamper events as evidence for the same.

In this chapter, we propose methods for detecting camera tampering events like camera occlusion, defocus and displacement in static surveillance cameras. A background model of the intended FOV is created by adopting a state of the art technique [35]. The method is slightly modified so as to make it robust for camera tamper detection. By using information from the background model created, the foreground objects present in newer/current frames are extracted. The area of foreground objects and dge details in newer frames are used as features

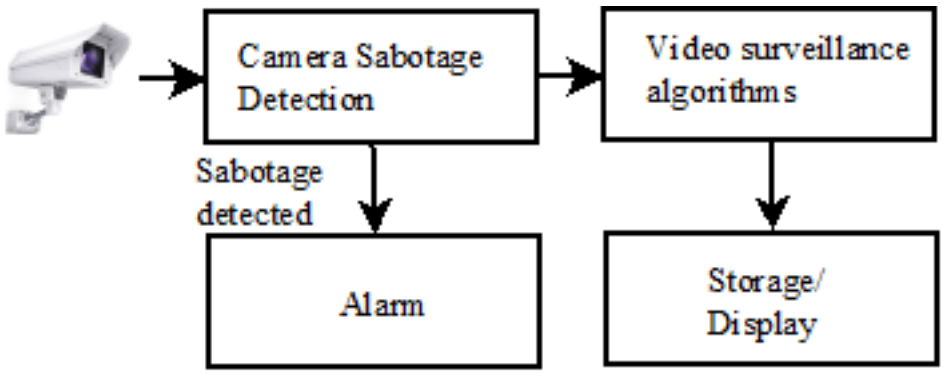


Figure 5.1: Block diagram of a video surveillance system with camera sabotage detection

for deciding camera tamper activity. An alarm is triggered for sustained events which is present in more than 12 frames to reduce false alarm rate.

The rest of this chapter is organized as follows. Background modeling is described in Section 5.1. Section 5.2 discusses the proposed method for camera tamper detection. Experimental results regarding performance of the proposed method is given in Section 5.3. Conclusion and recommendations on future research are summarized in Section 5.4.

5.1 Background Modeling

Local Illumination based Background Subtraction (LIBS) scheme [35] for background modeling and object detection from videos consists of two stages:

Stage 1 Identifying the stationary pixels in a frame sub-sequence taken for constructing the background model.

Stage 2 Comparison of newer frames to the created background model using local thresholding for finding the foreground objects in FOV.

In this section we provide a brief explanation of how it is performed as the method adopted for background estimation and object extraction is taken from [35]. Video frame in RGB are converted to gray-scale frames. The frames are scaled to half of its size after converting to gray-scale for reducing the computation time.

5.1.1 Background Model Creation

The background model should be robust to illumination variation, dynamic objects and small changes in FOV. For this, a sequence of frames of length are taken for identifying the stationary pixels in FOV. Let 'n' be the number of frames in the sequence. If background modeling is accurate, then foreground object extraction will give better results. This in turn increases the accuracy of proposed camera tamper detection system. Rather than choosing any random sequence of n frames in a video, one can take frames with illumination variation from previously recorded video sequences if possible. It provides a range of intensity values corresponding to each background pixel in the FOV. The value of n is between 20 to 30. If n is less, background estimation will not be effective. If $n > 30$, then time complexity of background estimation increases without much improvement in it. In real time applications, we have to perform background modeling in short time intervals to include new foreground objects which are static in FOV. Hence, the time complexity of background modeling should be reasonable.

The sequence of n frames under consideration is partitioned into overlapping sub-sequences having odd number of frames. Let 'W' be the number of frames in each sub-sequence. The mean of pixel values at location (x, y) in a sub-sequence of frames of W are then computed. Similarly, the mean of pixel values corresponding to each pixel location in W is computed. The mean of the values at the same pixel location (x, y) of all the frames in a sub-sequence starting at time t can be computed by,

$$m(x, y) = \frac{1}{W} \cdot \sum_{k=t}^{t+W} frame_k(x, y) \tag{5.1}$$

where $frame_k(x, y)$ indicates the pixel value at $(x, y)^{th}$ location of k^{th} frame in a sub-sequence. $m(x, y)$ is the mean of pixel values at location (x, y) in all of the frames in a sub-sequence of length W starting at time t. The pixel values at $frame_k(x, y)$ where $k \in [t, t + W)$ are kept in a 1-D vector V.

Hati et al. [35] used the standard deviation of intensity values in each sub-sequence for background modeling. We computed the mean of intensity values. It is observed that mean is providing better results in camera tamper detection phase.

The absolute differences between each element in V to the middle element in it is computed except for the middle one with itself. That is, the absolute difference between first and middle element in V is computed. Then, the absolute difference between second and middle element and so on is computed. These differences are stored in a 1-D vector D. It is performed for finding the variation in pixel values at each location in frame sub-sequences. The sum of $\left\lfloor \frac{W}{2} \right\rfloor$ lowest values in D are computed, sum_D. A pixel in $frame_{t+\lfloor \frac{W}{2} \rfloor}$ at position (x, y) is classified as stationary if the sum value is less than or equal to $\left\lfloor \frac{W}{2} \right\rfloor \cdot m(x, y)$.

$$sum_D(x, y) \leq \left\lfloor \frac{W}{2} \right\rfloor \cdot m(x, y) \tag{5.2}$$

The corresponding pixel is classified as non-stationary if Eq.(5.2) is not satisfied. For every pixel position (x, y) in the sub-sequence where $x \in [0, m_1 - 1]$ and $y \in [0, m_2 - 1]$, the condition in Eq. (5.2) is evaluated to classify the pixel location as stationary or non-stationary.

After the processing of all pixel positions in the current sub-sequence which starts at 't', the operations that performed on this sub-sequence is then performed on the next sub-sequence of W frames from 't+1'. This process is repeated on all possible sub-sequences of length W in the frame sequence of n frames taken for background modeling. m_1 and m_2 denotes the rows and columns respectively of the video frames under consideration. For each stationary pixel positions in the sequence from $\left\lceil \frac{W}{2} \right\rceil$ to $n - \left\lfloor \frac{W}{2} \right\rfloor$, two matrices M and N of size $m_1 \times m_2$ are used for keeping the minimum and maximum values in the range of pixel intensities respectively. This helps in identifying the foreground objects in FOV.

5.1.2 Object Extraction

For finding the foreground objects in current/newer frames, local thresholding based approach is used. A constant C is used for computing the lower and upper thresholds T_L and T_U respectively for local thresholding. For a pixel at position (x, y) in current frame, f, the values in M and N at corresponding positions are

used to compute the threshold values T, T_L and T_U as,

$$T(x,y) = \frac{1}{C}(M(x,y) + N(x,y)) \tag{5.3}$$

$$T_L(x,y) = M(x,y) - T \tag{5.4}$$

$$T_U(x,y) = N(x,y) + T \tag{5.5}$$

If the pixel intensity at (x,y) is between $T_L(x,y)$ and $T_U(x,y)$, then the corresponding pixel is classified as a background pixel.

$$T_L(x,y) <= f(x,y) <= T_U(x,y) \tag{5.6}$$

If Eq. (5.6) is satisfied, then it is a background pixel. If not, the pixel is classified as foreground. Background and foreground pixels are indicated by assigning 0 and 1 respectively. Each of the pixels in current frame are classified to background or foreground according to Eq. (5.6) and corresponding values are assigned.

The foreground object extraction method provide a binary image *obj_det* of size $m_1 \times m_2$ (same dimension as that of surveillance video frame size). That is, 0 and 1 for background and foreground pixels respectively. Due to noise, it is observed that certain pixels or small groups of pixels were wrongly classified as objects in *obj_det*. It may have negative impact on the proposed method for camera tamper detection. Hence, we eliminated them using the following decision strategy. The area occupied by each and every foreground objects in *obj_det* is computed separately. If the area occupied by any of these foreground objects is less than 8, then such foreground pixels are eliminated. Those pixels are converted to background. It helps in reducing the noise residues and false negatives.

5.2 Proposed Method

The flowchart of proposed method is shown in Fig. 5.2. Here, we discuss three types of camera tamper events: Camera defocus, camera occlusion and camera displacement. It starts with background modeling. *tod* and *tdf* are the counters kept for obtaining information on sustainability of camera tamper events. *tod* keeps track of the duration of camera occlusion or displacement events. And *tdf* keeps the duration of camera defocus event. We assume that at least 1 sec is

required for performing any criminal activities in FOV. Based on this assumption, an alarm is triggered for camera tamper events which have a duration of at least 0.5sec. This helps in reducing the false alarm rate of the proposed method. *tod* and *tdf* are initialized to 0. When the proposed method detects camera occlusion or displacement in current frame, f, the corresponding counter (i.e, *tod*) is incremented. Similarly, when the proposed method detects camera defocus, *tdf* is incremented. If *tod* and *tdf* reaches a particular threshold value Th_d, alarm is triggered for corresponding tamper event. After checking for camera tamper events, the frames which are free of any such events can be used for updating the background. This process of testing frames for camera tamper events continue until there are no new frames to process.

5.2.1 Camera Occlusion or Camera Displacement Detection

Camera occlusion leads to the inclusion of large foreground objects in FOV. Thus, the newer frames recorded during occlusion will be partially or totally distinct from the estimated background scene depending on the type of occlusion (partial or full). The newer frames recorded by the surveillance systems have to pass through object detection phase discussed in Sect.5.1 for identifying the foreground objects in it. This provide the binary image, *obj_det*. Now that, we can check for the presence of camera occlusion in newer frames. If any of the foreground object appears to be bigger than $(\frac{1}{3})^{rd}$ of the frame, then there is a chance that this object is suspicious. For obstructing the camera view, objects will be placed in front of the camera and very close to it. Such objects appear bigger than its actual size as it is placed close to camera lens. In *obj_det*, it occupies at least $(\frac{1}{3})^{rd}$ area of the frame. Thus, the size of each foreground object in f is taken as a parameter for camera occlusion detection. The area occupied by each of the foreground objects are computed (considering 8-neighborhood). If the area occupied by any of the foreground objects is greater than $(\frac{1}{3})^{rd}$ of the frame size of the surveillance video, then it indicates camera occlusion.

$$\frac{obj_{area}}{(m_1 \cdot m_2)} \geq Th_a \tag{5.7}$$

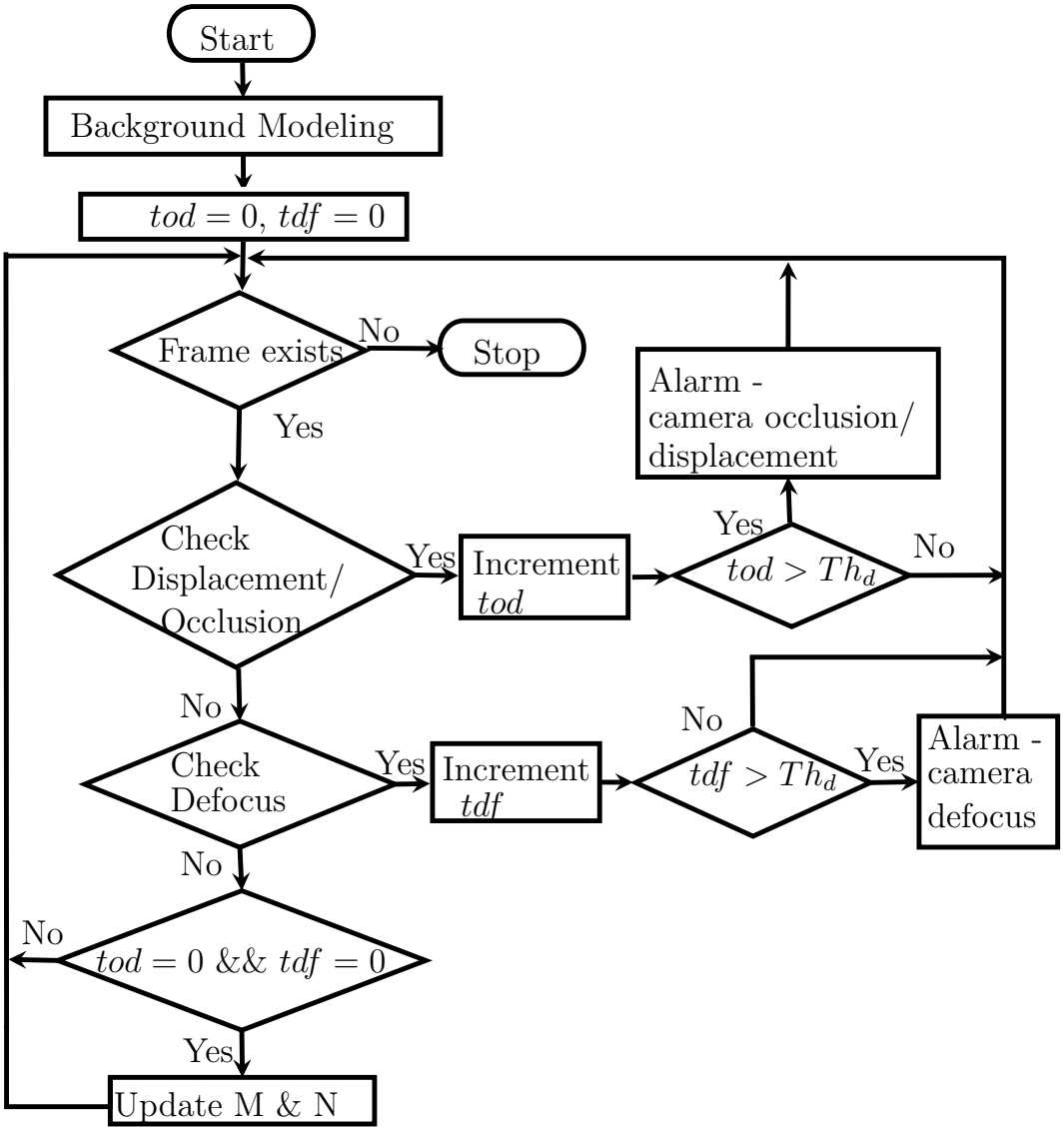


Figure 5.2: Flowchart of the proposed method for camera tamper detection in static surveillance systems

where Th_a is the threshold value which is set to $\frac{1}{3}$ and obj_{area} is the area occupied by a foreground object in obj_det.

Camera displacement will also produce new frames which are partially or totally distinct from the background scene. When a surveillance camera is moved physically, the FOV recorded by camera will change. The foreground object ex-

traction method in Sect.5.1 detect this change in view as foreground objects in FOV. This is because, the changed view recorded in newer frames will not be present in the estimated background model. A small physical displacement will considerable change in FOV. The object detection algorithm will return it as a large object in FOV. Hence, Eq. (5.7) can also be used for camera displacement detection. Figure 5.3 provides the flowchart of the proposed method for detecting camera occlusion or camera displacement in static surveillance systems. The results obtained from camera occlusion or camera displacement checking are returned to "Check Occlusion" decision box in Fig. 5.2.

We have to check for the persistence of these events to avoid false alarms that may occur due to crowd and large objects passing through the scene. If it is crowd, at least one person will make a movement in between as it is human nature. Hence, the object position or object boundary will differ in the previous, current and the next frames. The same concept is also applicable for moving large objects. For this, we compute the absolute difference between *obj_det* of previous and current frame denoted as $diff_p$. The same is computed between the *obj_det* of current and next frame denoted as $diff_n$. If the object is moving, then the movement will be recorded in difference images $diff_p$ and $diff_n$. As *obj_det* is a binary image, $diff_p$ and $diff_n$ are also binary images. If the object is stationary, then the number of non-zero elements in $diff_p$ and $diff_n$ is 0 or almost 0 (considering noisy pixels). The number of non-zero elements in $diff_p$ and $diff_n$ are computed, denoted by $diff_{p_{count}}$ and $diff_{n_{count}}$. If these values are less than a threshold value Th_3, it indicates camera tampering.

$$\frac{diff_{p_{count}}}{(m_1 \cdot m_2)} \leq Th_3 \quad \&\& \quad \frac{diff_{n_{count}}}{(m_1 \cdot m_2)} \leq Th_3 \tag{5.8}$$

To reduce computation, we skip one frame at a time. i.e, after processing frame t, the next frame taken for camera tamper detection is $t + 2$. While processing frame 't', its immediate past and future frames are also processed for identifying the tamper activity. Hence, avoiding the repetition will reduce computation time. As we skip one frame on each iteration, Th_d in Fig. 5.2 is set to 6 which ensures that the suspicious camera tamper event is present in at least 12 continuous frames (for videos of 25 fps).

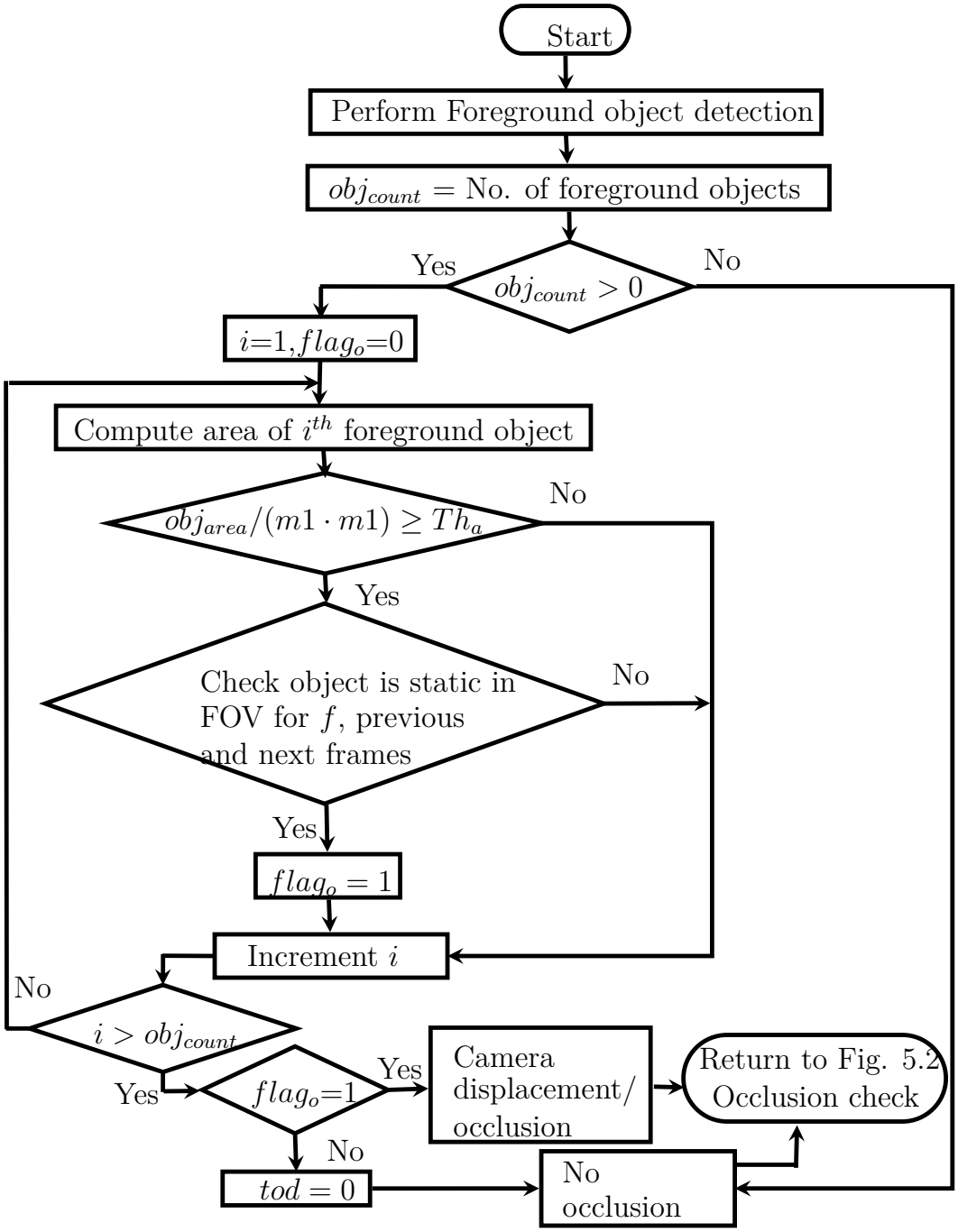


Figure 5.3: Flowchart of the proposed method for detecting camera occlusion or camera displacement in static surveillance systems

5.2.2 Camera Defocus Detection

Camera defocus leads to reduced visibility, which in turn leads to less edge pixels in recorded frames. Edge pixels are computed from background model M and current frame f for its detection. If the number of edge pixels, f_{edge_count}, in f is less than a threshold times that in the background M_{edge_count}, it indicates camera defocus, concept borrowed from [4].

$$\frac{f_{edge\_count}}{M_{edge\_count}} \leq Th_{sdf} \tag{5.9}$$

The condition check in Eq. (5.9) will detect camera defocus tamper event. Th_{sdf} is the threshold value. Figure 5.4 gives flowchart of the proposed method for detecting camera defocus anomaly. The result obtained from camera defocus detection method is returned to "Check Defocus" decision box in Fig. 5.2.

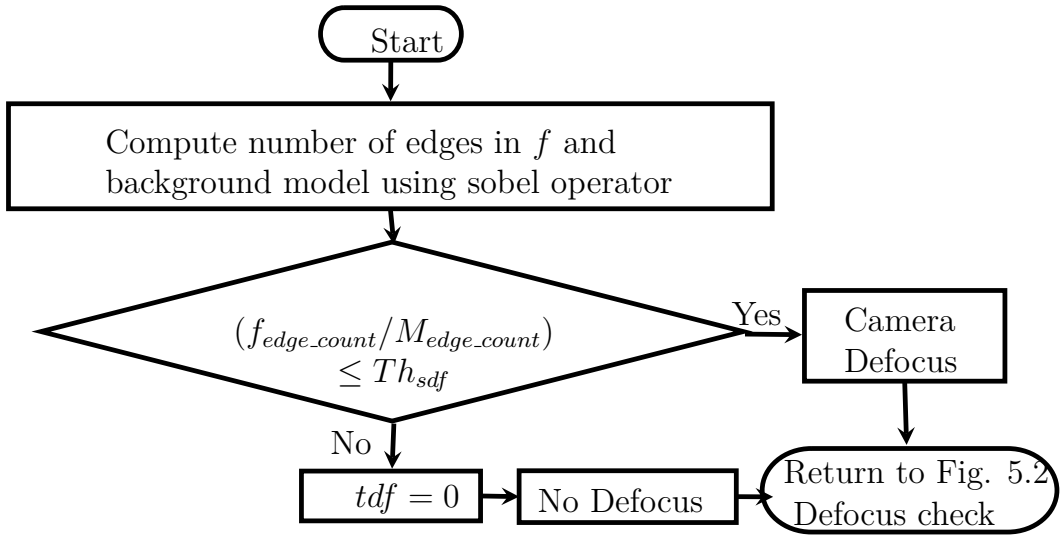


Figure 5.4: Flowchart of the proposed method for detecting camera defocus in static surveillance systems

5.3 Experimental Results and Discussion

The proposed method is implemented in MATLAB R2013a(8.1.0.604), performed on a 2.50 GHz Intel Core i5 PC. The parameters W, C, Th_{sdf} and Th_1, and are

empirically set to 9, 9, 0.7, and 0.02 respectively.

Experiments are conducted on recorded videos simulating various challenging situations and also on the camera tampering detection dataset provided by Saglam and Temizel [91]. The details of the dataset provided by Saglam and Temizel [91] are discussed in Subsection 2.7.3 of Chapter 2.

5.3.1 Camera Tampering Dataset Created

We created 10 test videos corresponding to each camera anomaly (namely camera occlusion, camera defocus and camera displacement) totaling 30 videos. These are HD (1280×720) resolution videos recorded at 25 fps. To have an idea on false alarms, 6 surveillance videos without camera tamper events are taken from i-LIDS dataset [43]. We captured 10 non-tamper videos, totaling to 16 videos without any camera tamper events. The details regarding number of videos in each camera tampering category is given in Table 5.1.

Table 5.1: Summary of static camera tampering dataset created by us

Tamper Type	**Number of videos**
Defocus	10
Occlusion	10
Displacement	10
No tamper [43]	6

The frames taken from some of these experimental videos are shown in Fig. 5.5 and 5.6. Figure 5.5 (a), (b) and (c) correspond to the intended FOV frames recorded under the normal scenario. Figure 5.6 (a), (b) and (c) are another set of frames showing intended FOVs. Camera views recorded after turning the camera away from the intended FOVs are shown in Fig. 5.5 (d) and Fig. 5.6 (d). Camera defocus by spraying aerosol is given in Fig. 5.5 (e). Camera defocus by placing water drop on camera lens is given in Fig. 5.6 (e). Camera occlusion by placing

opaque objects adjacent to surveillance camera lens (in front of the camera) is shown in Fig. 5.5 (f) and Fig. 5.6 (f).

Figure 5.5: Sample frames from our dataset showing visuals of camera tamper events and their intended FOVs in static surveillance systems. (a), (b) and (c) are frames showing intended FOVs. (d) Scene recorded during camera displacement. (e) Scene recorded during camera defocus. (f) Scene recorded during camera occlusion.

Figure 5.6: Sample frames from our dataset showing visuals of camera tamper events and their intended FOVs in static surveillance systems. (a), (b) and (c) are frames showing intended FOVs. (d) Scene recorded during camera displacement. (e) Scene recorded during camera defocus. (f) Scene recorded during camera occlusion.

5.3.2 Performance Evaluation

The performance of proposed method is evaluated in terms of precision (P), recall (R), false positive rate (FPR) and accuracy (A). FPR is also called as false alarm rate. P, R, and A are computed using Eq. 2.2, Eq. 2.1 and Eq. 2.3 respectively (discussed in Sub-section 2.8 of Chapter 2). FPR is computed using Eq. 3.15 (Sub-section 3.3.2 of Chapter 3). Here, the description of TP, FN, TN and FP are as follows when camera tampering is considered as the positive class:

- **TP** - Alarm triggered for an actual camera tamper event.
- **TN** - Camera non-tamper events categorized as non-tamper.
- **FP** - Non-tamper event wrongly categorized as tamper event.
- **FN** - Camera tamper events categorized as non-tamper.

Among 30 camera tamper videos in our dataset, the proposed method missed 1 event in camera defocus and 1 in camera occlusion. Camera displacement events are detected successfully. No alarm is triggered for non-tamper events. P, R, FPR and A of the proposed method on our dataset are 100%, 93.33%, 0% and 95.65% respectively.

A comparison on the performance of the proposed method with works in [55, 91, 122] is shown in Table 5.2. Dataset provided by Saglam and Temizel [91] is used for comparison. We received only 39 videos out of the original 54 sequences used in [91]. In [91], Saglam and Temizel compared their work with that in [4, 30, 87] on the same dataset. From this, we can say that the proposed system has high precision and low false alarm rate. Updating the background model with background information from recent frames in specific time interval will keep the system robust.

Our method may fail in following situations – 1) object used for camera occlusion is moving/shaking rapidly and (2) scene background containing highly smooth regions with very less edge components. The proposed system failed to detect 4 camera tamper events - 1 from camera displacement and 1 from camera defocus and 2 from camera occlusion. The frames corresponding to tamper

events caused by camera defocus and displacement anomalies and their corresponding intended FOVs are shown in Fig. 5.7. The frames in Fig. 5.7 (a) and (c) are intended FOVs. Figure 5.7 (b) corresponds to the frame recorded when camera defocus is employed on the intended FOV in Fig. 5.7 (a). Figure 5.7 (d) corresponds to the frame recorded during camera displacement anomaly when employed on the intended FOV in Fig. 5.7 (c). The intended FOV frames in these cases have highly uniform areas, and therefore, our method failed. These kinds of FOVs rarely occur in reality.

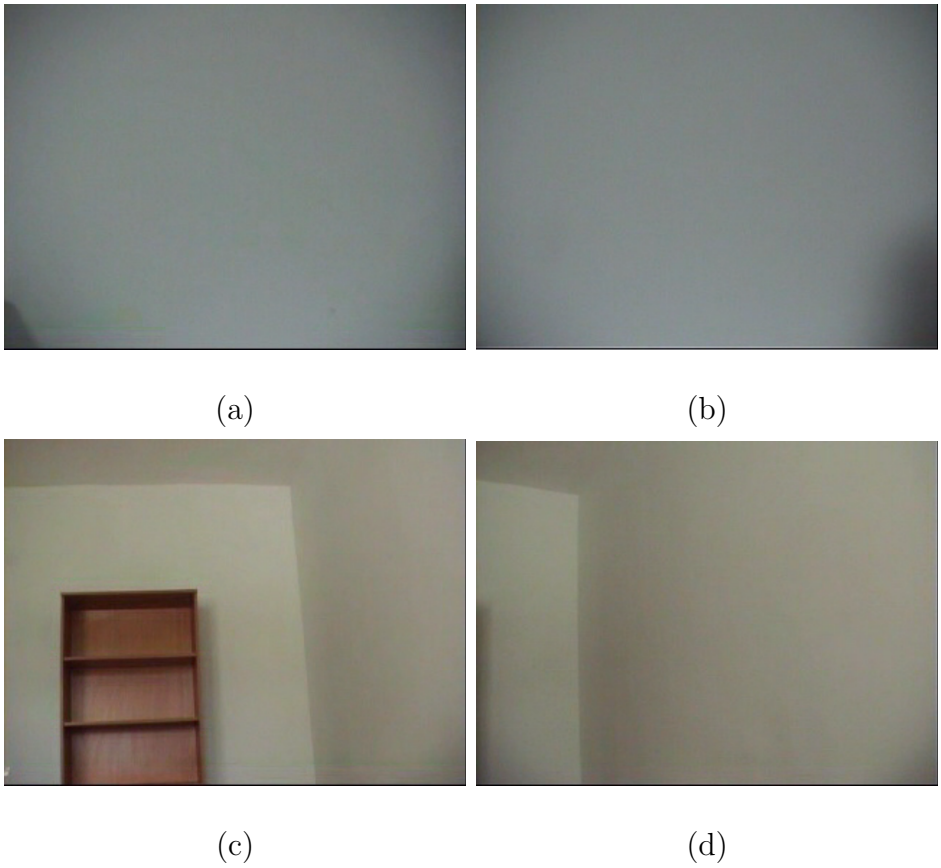


Figure 5.7: Sample frames from static surveillance dataset in [91] showing visuals of camera tamper events and their intended FOVs where the proposed method failed. (a) and (c) are frames showing intended FOVs. (b) Scene recorded during camera defocus from the intended FOV in (a). (d) Scene recorded during camera displacement from the intended FOV in (c).

Table 5.2: Performance comparison of the proposed method with existing methods

Tamper Type	Different Approaches	Results			
		Tamper events	True Positives	False Negatives	False Positives
Defocus	Gil-Jiménez et al. [30]	35	22	13	1
	Aksay et al. [4]	35	28	7	5
	Ribnick et al. [87]	35	21	14	0
	Saglam and Temizel [91]	35	29	6	0
	Tsesmelis et al. [122]	13	11	2	5
	Kryjak et al. [55]	14	14	0	0
	Proposed method	**13**	**12**	**1**	**0**
Occlusion	Gil-Jiménez et al. [30]	40	35	5	14
	Aksay et al. [4]	40	33	7	2
	Ribnick et al. [87]	40	37	3	1
	Saglam and Temizel [91]	40	38	2	0
	Tsesmelis et al. [122]	17	17	0	2
	Kryjak et al. [55]	11	11	0	4
	Proposed method	**17**	**15**	**2**	**0**
Displacement	Gil-Jiménez et al. [30]	12	8	4	3
	Ribnick et al. [87]	12	7	5	0
	Saglam and Temizel [91]	12	11	1	0
	Tsesmelis et al. [122]	12	12	0	3
	Kryjak et al. [55]	12	12	0	2
	Proposed method	**12**	**11**	**1**	**0**

The methods in [91], [136], [55], [64] and [39] can process video frames with resolutions of VGA@12.8 fps, VGA@20.8 fps, VGA@60 fps, 320×240@26 fps and 320×240@30 fps respectively. The proposed method can process video frames with resolutions 320×240, VGA and HD on an average of $60 - 70$ fps, $25 - 30$ fps and $10 - 13$ fps respectively. This shows that our method is computationally efficient compared to other methods and can be implemented for real-time processing. The proposed method could also detect fogging, as fog causes reduced visibility.

5.4 Summary

A simple and efficient method for camera tamper detection due to camera defocus, occlusion and displacement in static surveillance systems is presented. The algorithm is based on background modeling, edge components, foreground object's area, its relative position in current, previous and future frames.

Experimental results on video sequences containing tamper events and non-tamper events demonstrate the effectiveness of the proposed system. Videos having 640×480 pixel resolution are processed at $25 - 30$ fps. Hence, the proposed method can be implemented in real-time with high precision and low false alarm rate.

Future work could concentrate on improving the precision of the proposed method on smooth background surveillance scene and camera occlusion using rapidly moving/shaking objects. Extending this work to panning surveillance cameras are also under consideration.

Chapter 6

Camera Tampering Detection in PTZ Video Surveillance Systems

Active or pan-tilt-zoom (PTZ) surveillance cameras are prevalent nowadays as opposed to static surveillance cameras. PTZ camera can cover a large FOV. PTZ cameras can automatically operate pan, tilt, and zoom in a preset backdrop scene for tracking the foreground objects. Normally, PTZ cameras are entitled to perform panning operation in an intended FOV. The operator-in-charge of surveillance system performs zoom and tilt operations after successful authentication via the display and control unit attached to it. Here, we consider a PTZ camera with this settings.

A mosaic view of the whole panning expanse under surveillance is designed as a background reference model. The incoming/present frames are registered and matched to the reference for extracting the foreground objects. SIFT [68] key point conformities using homography are employed for registering the present frames to the background mosaic. The area and static characteristics of foreground objects, edge information and the count of frames recorded per camera pan cycle are the features for identifying questionable camera tamper events. To lessen false positives, an alert is triggered for prolonged camera tamper events only.

The rest of this chapter is organized as follows. Section 6.1 deals with background modeling and foreground object extraction in panning surveillance systems. Section 6.2 describes the proposed method for camera tamper detection.

Section 6.3 deals with experimental results and discussion. Conclusions and future research directions are presented in Section 6.4.

6.1 Background Modeling

The background modeling method discussed in [122] can be adopted here. Tsesmelis et al. [122] stores several frames for covering the whole panning area in a database so that the newer frames can get a match from one of the images in the database. This mapping is accomplished by employing Fast Library for Approximate Nearest Neighbors (FLANN) [77] matching algorithm. To lessen the number of frames stored, frames that overlap at half frame width are taken. The panning movement is asynchronous with its frame rate in an active camera. Hence, the frames in the following panning sweep may not find an exact match frame from the saved backdrop frames. Further, the frames recorded in between the half-frame-width distance may contain a few regions that may not exist in its most matched background frame. Therefore, it is intuitive to keep a mosaic view of the complete pan range covered by the PTZ surveillance system as the background model.

6.1.1 Mosaic View Creation

The distance separating consecutive frames is obtained using Rigid Transformation [115]. $2-D$ Rigid transformation estimated from two consecutive frames ($(n-1)^{th}$ and n^{th}) is a 3×3 matrix, R, where $R(1,3)$ and $R(2,3)$ correspond to the translations along X and Y axes respectively. The sign of $R(1,3)$ and $R(2,3)$ symbolizes the direction of movement of the camera. A sign change in these symbolizes that the camera has started moving in reverse direction. This aids in locating the limits of panning area.

The mosaic view (*mosaic*) is initialized with the boundary frame. *mosaic* is later updated with frames that are overlaying by half frame width. If we take consecutive frames for *mosaic* creation, it increases time complexity without contributing much information regarding the background. To find a frame, j, which is at half frame width distance from the earlier frame, i, used for mosaic

update, R between consecutive frames that provides the horizontal and vertical displacements are computed. $R(1,3)$ values (if the direction of panning is horizontal, otherwise $R(2,3)$) estimated from a consecutive pair of frames are added up (denoted as sum) by,

$$sum = \sum_{k=i+1}^{j} R_k(1,3) \tag{6.1}$$

When the aggregated sum in Eq.(6.1) reaches half frame width (for vertical panning, frame height is used), the current frame is chosen for updating the background mosaic view and sum is reinitialized to 0.

$$sum \geq frame_{width}/2 \tag{6.2}$$

Mosaic updation using incoming frames that satisfy Eq. (6.2) is continued till the next panning boundary frame. The count of frames recorded between these panning boundaries are saved as a reference for total possible count of frames (tf_{count}) recorded in a pan cycle. *mosaic* should be updated with boundary frames, irrespective of how distant it is from the last frame used for updating *mosaic*. Otherwise, *mosaic* may not contain the border regions of an intended FOV.

vl_feat mosaic algorithm [126] is used for creating mosaic of two images under consideration. vl_feat *mosaic* algorithm is executed each time with existing *mosaic* and current frame when the condition in Eq.(6.2) is satisfied. The concepts and applications of video mosaics are explained in [115]. In vl_feat *mosaic* algorithm [126], SIFT key points extracted from the images to be stitched together are processed to find the matching key points existing in these images. The homography matrix, H, of size 3×3 between the images are then computed using any 4 key points from the set of matched key points. For better image registration, they used RANdom SAmple Consensus algorithm (RANSAC) [27] for random selection of matched key points to find the best H. This is done by computing the error between the projection of key points in first image with their corresponding points in second image. All matches which have error less than a predetermined threshold are considered as inliers and rest are outliers. The quality of a predicted H depends on the number of inliers. The best H is the one which have more inliers in an iteration of 100 with 4 random samples from

the matched key points. Using H, one image is warped to be in the same frame as the other. A new image initialized with 0 is created for stitching both images. The pixel values in the non-overlapping regions of the first and second images are copied as it is to the new frame and average of pixel values are taken for overlapping pixels positions.

The background mosaic generated from an example video having the visuals of a parking lot taken from the dataset in [122] is presented in Fig. 6.1. Algorithm 10 provides the procedure for creating *mosaic*. The condition in step 14 is for locating the border frame. It is also used for reducing the error in background modeling that may arise because of camera calibration. To boost the processing speed, we skip 1 frame at a time. Therefore, *sum* need to be updated with twice the distance estimated between adjacent frames.

Figure 6.1: Background mosaic created for the parking lot video from the dataset in [122]

Locating boundary frames using sign change will not work when the camera is rotating 360°. The frame captured by the surveillance camera at the time of background *mosaic* initialization can be saved as an initial frame. The steps for background modeling are followed till it reaches a frame that is highly correlated to the initial frame.

6.1.2 Foreground Object Extraction

The FOV present in an incoming frame is only a part of the intended FOV. Hence, these frames have to be registered on to *mosaic* to identify the foreground

Algorithm 10 Procedure for Background mosaic modeling

1: $sum \leftarrow 0$, $flag \leftarrow 0$;
2: Initialize *mosaic* with border frame;
3: Read next frame to cur_f;
4: Compute H between cur_f and previous frame;
5: Assign sign of $H(1,3)$ to cx_{sign};
6: **if** $flag = 0$ **then**
7: $prev\_x_{sign} \leftarrow cx_{sign}$;
8: $flag \leftarrow 1$;
9: **else**
10: **if** $prev\_x_{sign} = cx_{sign}$ **then**
11: $sum \leftarrow sum + 2 \cdot H(1,3)$;
12: **else**
13: Find sign of $H(1,3)$ between cur_f and 2 immediate future frames and assign it to $n1\_x_{sign}$ and $n2\_x_{sign}$ respectively;
14: **if** $cx_{sign} = n1\_x_{sign}$ and $cx_{sign} = n2\_x_{sign}$ **then**
15: Update *mosaic* with information from cur_f;
16: Set $sum \leftarrow 0$;
17: **else**
18: $sum \leftarrow sum + 2 \cdot H(1,3)$;
19: **end if**
20: **end if**
21: **end if**
22: **if** $sum \geq frame\_width/2$ **then**
23: Update *mosaic* with information from cur_f;
24: Set $sum \leftarrow 0$;
25: **end if**
26: Set $prev\_x_{sign} \leftarrow cx_{sign}$;
27: Repeat from step 3 to step 25 until the next border frame;

objects in it. SIFT feature descriptors [68] from *mosaic* and current frame (cur_f) are extracted for identifying the matching key points. Using RANSAC, the best homography, H, is obtained. Before proceeding to foreground extraction, we need

to examine whether the estimated homography is valid.

Using SVD, H is decomposed into three 3×3 matrices - 2 orthogonal matrices U and V, and 1 pseudo-diagonal matrix S. H is a valid homography if it satisfies the following two conditions.

1. The diagonal elements $S(1,1)$ and $S(2,2)$ should be real positive numbers as they represent the singular values of H.

2. The ratio between these two values should be less than a threshold value.

$$\frac{S(1,1)}{S(2,2)} < Th \tag{6.3}$$

If estimated H is valid, then the cur_f is registered on to *mosaic*. A training stage is carried out before *mosaic* creation to identify the possible values of Th by estimating H between consecutive frames. The maximum value obtained on performing $S(1,1)/S(2,2)$ is assigned to Th.

The foreground objects in cur_f are extracted using local thresholding. The upper and lower thresholds are computed using N, M, and C. N and M are initialized with *mosaic*. They are updated using information from incoming frames after examining them for camera tampering detection. C is a constant. For each pixel (x, y) at the intersecting region between cur_f and *mosaic*, the threshold values $T_L(x, y)$, $T_U(x, y)$, and $T(x, y)$ are computed as:

$$T(x,y) = \frac{1}{C}(M(x,y) + N(x,y)) \tag{6.4}$$

$$T_L(x,y) = M(x,y) - T(x,y) \tag{6.5}$$

$$T_U(x,y) = N(x,y) + T(x,y) \tag{6.6}$$

A pixel in cur_f having intensity value between upper and lower thresholds is classified as a background pixel. These background pixels are assigned 0 in its corresponding location in the object detection matrix, obj_{detect}.

$$T_L(x,y) \leq cur_f(x,y) \leq T_U(x,y) \tag{6.7}$$

If Eq. (6.7) does not hold, then the corresponding pixel is classified as a foreground pixel and is indicated with 1 in obj_{detect}.

Some pixels are misclassified as foreground pixels because of noise. For reducing the false alarms in camera tamper detection, we employed the method in [100]. Foreground objects having an area less than 8 are eliminated. The procedure for foreground object extraction is presented in Algorithm 11.

Algorithm 11 Procedure for foreground object extraction

Set $T \leftarrow (M + N)/C$;
$T_L \leftarrow M - T$;
$T_U \leftarrow N + T$;
for each pixel (x, y) in the registered region of cur_f to *mosaic* **do**
    if $T_L(x, y) \leq cur_f(x, y) \leq T_U(x, y)$ **then**
        $obj_{detect}(x, y) \leftarrow 0$;
    else
        $obj_{detect}(x, y) \leftarrow 1$;
    end if
end for

If the estimated H is not valid, it points to a situation where the camera may be tampered. A discussion on this condition is included in Subsection 6.2.1.

6.1.3 Updating the Background Mosaic

The background model is frequently updated to keep track of changes in the background. Extremely dynamic background objects (like fluttering leaves), illumination variations, changes in background geometry (like parked cars) and so on have to be taken care of. If the background model cannot hold these changes, it may increase the false alarms in camera tamper detection. N and M are employed for local thresholding in Subsection 6.1.2. They are maintained for incorporating the background variations. N and M are updated with intensities of background pixels in cur_f if it is classified as a normal frame without any camera tamper events. If $M(x, y) > cur_f(x, y)$, $M(x, y)$ is updated with $cur_f(x, y)$. $N(x, y)$ is updated with $cur_f(x, y)$ if $N(x, y) < cur_f(x, y)$. For each background location (x, y), $N(x, y)$ and $M(x, y)$ hold the highest and lowest intensities appearing at that location respectively. The static foreground objects (non-suspicious) present

in normal frames are also added to *mosaic* if they persist in the intended FOV for more than 2 minutes.

6.2 Proposed method for Camera Tamper Detection

The camera tampering detection method described in this chapter focuses on: defocus, occlusion, displacement. The proposed scheme could also recognize natural camera tampering events such as dirt on the camera lens, careless maintenance by the cleaning staff, fog and smoke. Careless maintenance activities may change the intended FOVs of surveillance cameras leading to camera displacement or occlusion detection. The change in an intended FOV because of the camera tampering activities may contribute foreground objects for those regions which are absent in the stored background model. Smoke, fog, and dirt may reduce the visibility or cause occlusion. Therefore, these conditions can be detected by the methods presented for camera defocus and occlusion. The flowchart of the proposed method for camera tampering detection is presented in Fig. 6.2. tdf, tpo, tdp, and tfo are the counters for reducing false positives. tdf is used for defocus, tpo for partial occlusion, tdp for displacement, and tfo for full occlusion anomalies. An alarm is activated for prolonged camera tamper events. tdf, tpo, tdp, and tfo are initialized to 0. On detecting any camera tamper anomaly, we increment the counter corresponding to the camera tamper event identified to keep track of the duration (repetition) of the same anomaly. A thresholding approach based on the counter value is used for identifying sustaining camera anomalies. An alarm associated with the identified camera tamper event is then activated. If the values of tdf, tpo, tdp, and tfo remain 0 even after processing a frame for camera tamper detection, then it indicates a non-tamper (normal without any camera tampering) frame. Such frames can be used for updating the background in N and M. The processing of incoming frames for detecting camera tamper events continue till there are no new frames to process.

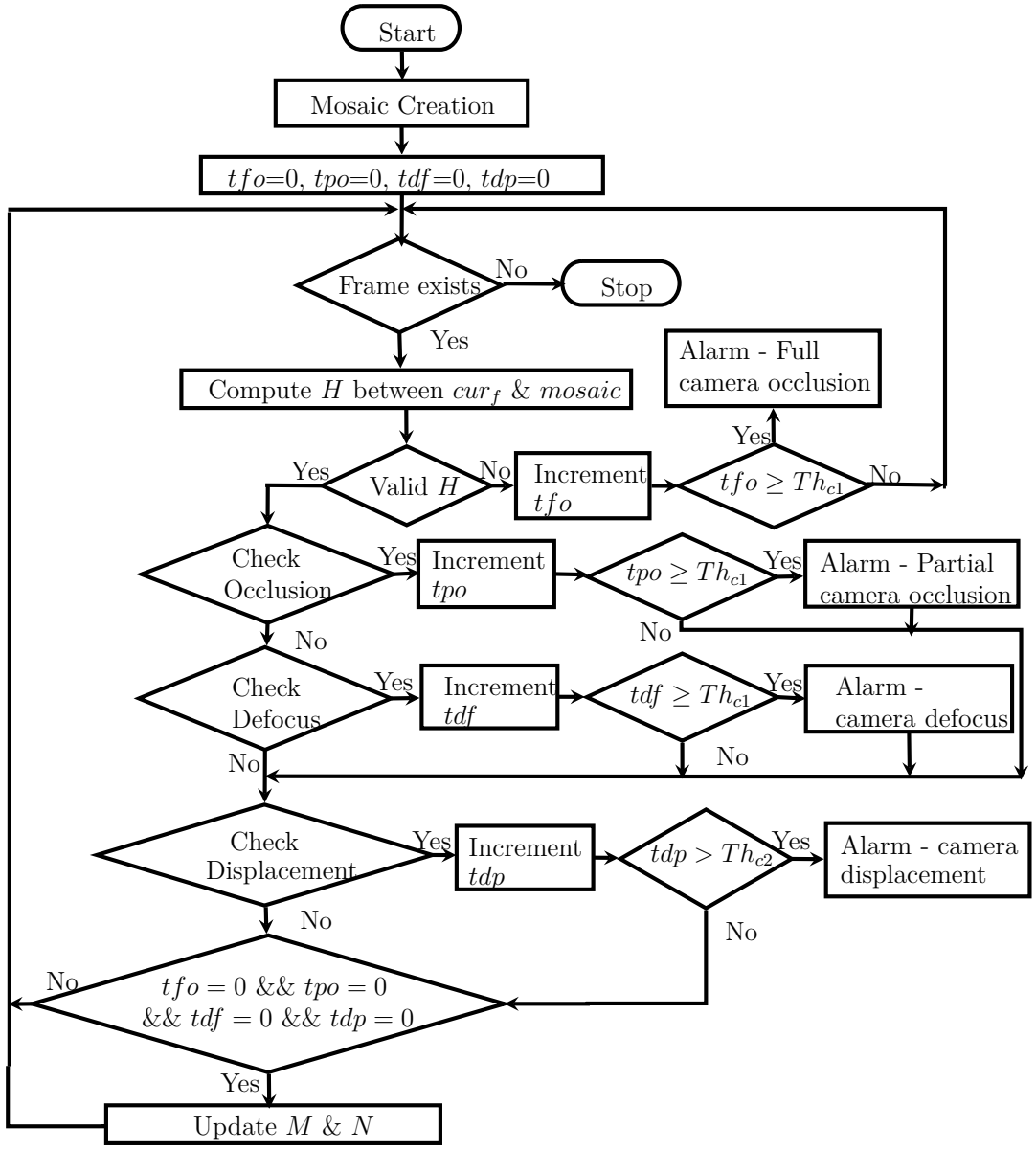


Figure 6.2: Flowchart of the proposed method for detecting camera tamper events in panning surveillance systems

6.2.1 Camera Occlusion Detection

Camera occlusion can be either full or partial. The intended FOV is completely obstructed in full occlusion where as a portion of the FOV is not recorded properly

in partial occlusion. In a panning surveillance system, partial occlusion can occur in two ways.

Case 1 An object placed adjacent to the camera and in front of it, so that the panning stretch corresponding to the object's position is occluded in intended FOV. The object used for occlusion is static in the intended FOV. As the camera is panning, this occluding object will not appear in all of the frames captured per camera pan cycle. We have to estimate the position where the occluding object is appearing in the intended FOV and the area of occlusion caused by this object for identifying the camera tamper event. The position is computed for determining the static nature of occluding objects. The foreground objects present in incoming frames are extracted using the method discussed in Section 6.1.2. When an object is placed adjacent to the camera lens, it seems larger than its actual size and occupies significant expanse in the frames captured. This is due to its nearness to the camera lens. Foreground objects occupying area (Obj_{area}) larger than $\frac{1}{4}^{th}$ of the frame size is considered as a suspicious object for camera occlusion.

$$\frac{Obj_{area}}{(m_1 \cdot m_2)} \geq Th_a \tag{6.8}$$

where m_1 and m_2 denote frame width and frame height respectively. As we concentrate on objects occupying large area which forms $\frac{1}{4}^{th}$ of the frame size, the value of Th_a in Eq. (6.8) is chosen to be 0.25. After recognizing the doubtful object, it is analyzed to verify if it is immobile in the intended FOV. The suspicious object present in an incoming frame (cur_f) is mapped on to the background mosaic model for identifying the pixel positions where it appears in *mosaic* and are stored in $cmosaic_{pxls}$ for later processing. Likewise, the pixel positions in *mosaic* occupied by the same object corresponding to its presence in the frames that immediately precedes ($pmosaic_{pxls}$) and follows ($nmosaic_{pxls}$) cur_f are also recorded. If these pixels $pmosaic_{pxls}$, $cmosaic_{pxls}$, and $nmosaic_{pxls}$ are coinciding, then the suspicious object is immobile in the intended FOV. The number of coinciding pixels in $pmosaic_{pxls}$ and $cmosaic_{pxls}$ is denoted as mpl_p and the

same between $nmosaic_{pxls}$ and $cmosaic_{pxls}$ is denoted as mpl_n. As the camera is panning and the suspicious object is immobile in the intended FOV, it is obvious that the area occupied by the occluding object in consecutive frames is varying. mpl_p and mpl_n are normalized and thresholded as:

$$\frac{mpl_p}{\min(sobj_{areap}, sobj_{areac})} \geq Th_{o1} \tag{6.9}$$

$$\frac{mpl_n}{\min(sobj_{areac}, sobj_{arean})} \geq Th_{o1} \tag{6.10}$$

where $sobj_{arean}$, $sobj_{areap}$, and $sobj_{areac}$ denote the area occupied by the occluding object in future, previous and current frames respectively. As mpl_p denote the tally of coinciding pixels, it cannot take a value which is higher than the lowest among $sobj_{areac}$ and $sobj_{areap}$. Likewise, mpl_n cannot be larger than the lowest among $sobj_{areac}$ and $sobj_{arean}$. If the quotient of division in Eq. (6.9) and Eq. (6.10) are ≈ 1, it symbolizes that the occluding object is present at the same location in *mosaic* for the present frame and its direct neighbors. Therefore, the threshold Th_{o1} should be ≈ 1 and is empirically fixed to 0.98. If Eq. (6.9) and Eq. (6.10) are satisfied, then we infer it as a tamper event caused by partial camera occlusion.

Case 2 To limit the view of recording, an object is placed on the camera itself. In an incoming frame, the foreground objects satisfying Eq. (6.8) are regarded as suspicious. As the object is placed on the camera, it occupies the same amount of area in each frame. These frames when mapped on to the background model shows that the suspicious object is not static in the FOV. But, it is static with respect to the camera. With this type of tampering, the border regions of an intended FOV can be occluded easily. To determine the pixels occupied by this suspicious object, cur_f and the frames that immediately precedes and follows it are registered on to *mosaic*. The coordinates of these pixel in *mosaic* corresponding to the current frame and the frames that immediately follows and precedes cur_f are stored as $cmosaic_{pxls}$, $nmosaic_{pxls}$ and $pmosaic_{pxls}$ respectively. To confirm the static nature of the object with respect to the camera, the transformation (or shift) between cur_f and the frame that immediately precedes it is

computed. The homography thus estimated is applied on $cmosaic_{pxls}$ to perform the reverse transformation. The number of coinciding pixels between $pmosaic_{pxls}$ and the transformed $cmosaic_{pxls}$ is represented as fpl_p. Likewise, fpl_n between cur_f and the frame that immediately follows is also computed. fpl_n and fpl_p are normalized and thresholded as:

$$\frac{fpl_p}{\min(sobj_{areap}, sobj_{areac})} \geq Th_{o2} \tag{6.11}$$

$$\frac{fpl_n}{\min(sobj_{areac}, sobj_{arean})} \geq Th_{o2} \tag{6.12}$$

where Th_{o2} denotes the threshold value. If the result of the division is ≈ 1, it symbolizes that the suspicious object is appearing at the same location in cur_f and the frames that immediately precedes and follows it. Hence, Th_{o2} must be ≈ 1. Considering the possible errors on applying transformation, Th_{o2} is empirically set to 0.96. If the conditions in Eq. (6.11) and Eq. (6.12) are satisfied, then we infer it as a tamper event caused by partial camera occlusion.

When the partial camera occlusion discussed in case 1 or case 2 happens, we have to confirm whether it is a prolonged event or not. Otherwise, the system may activate false alarms in situations where big objects or crowd crossing through the FOV. An alarm is activated on detecting case 1 or case 2 occlusion, if it continues to appear in 24 successive frames ($\approx$ 1 sec). As the presence or absence of a suspicious object in frames at position $(n + 1)$ and $(n - 1)$ are evaluated while handling the same on an n (current frame), we skip 1 frame and resume camera tampering detection from $(n + 2)$. This strategy decreases time complexity by eradicating the processing of a particular frame twice. Hence, the threshold Th_{c1} in the flowchart provided in Fig. 6.2 is assigned a value of 12. The flowchart of the proposed method for detecting camera occlusion is presented in Fig. 6.3. The outcomes of a camera occlusion detection test are returned to the "Check Occlusion" decision box in Fig. 6.2.

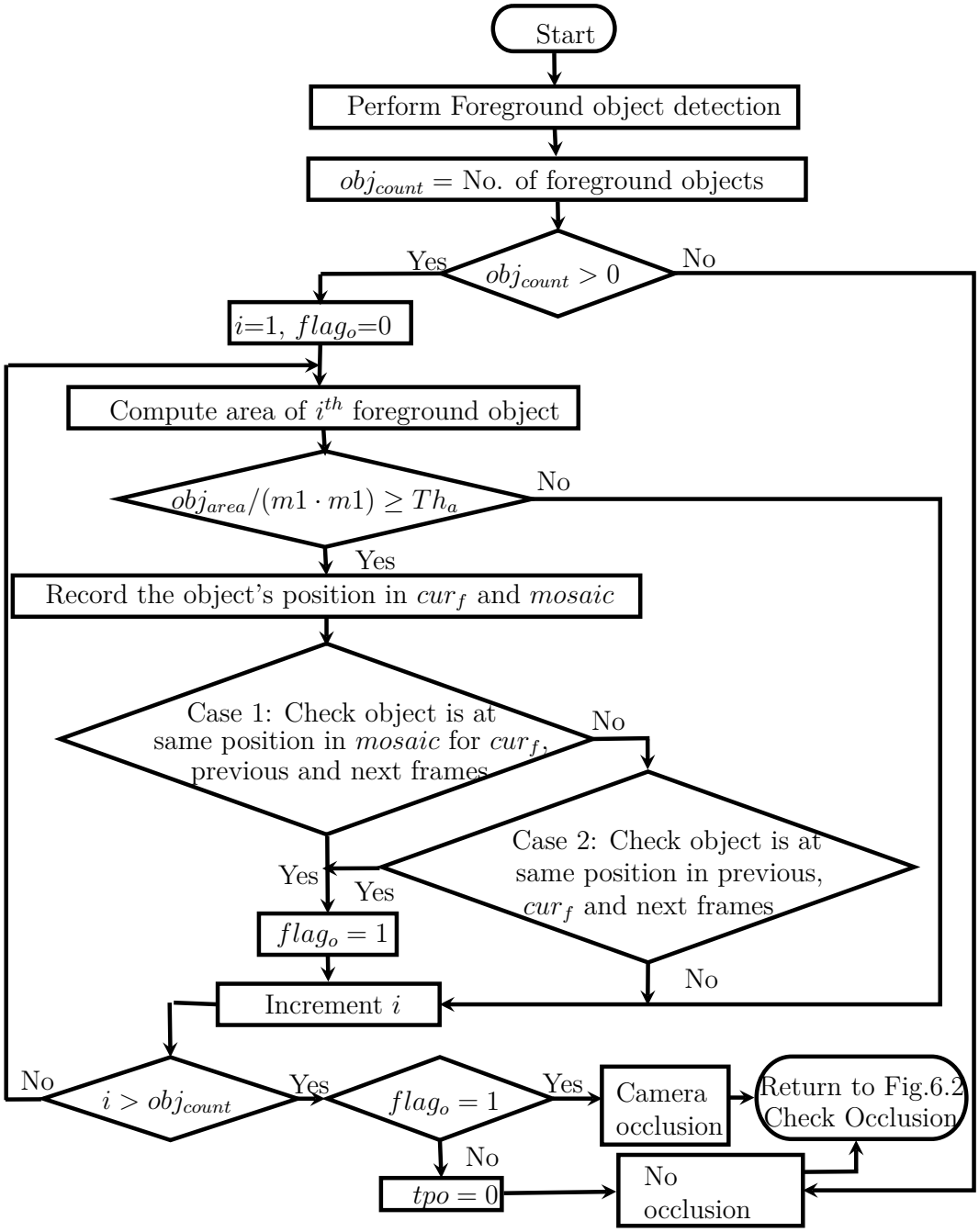


Figure 6.3: Flowchart of the proposed method for detecting camera occlusion in panning surveillance systems

The methods for partial camera occlusion detection fail in full occlusion. The system fails to register the incoming frames captured during full occlusion to the background *mosaic*. Homography, H evaluated from such a frame is invalid. These frames lack the visuals corresponding to an intended FOV. Hence, there is no match between *mosaic* and cur_f. It indicates that the visuals of cur_f is not an intended FOV which is to be classified as a camera tamper event. As in partial occlusion, an alarm is activated if this anomaly continues to exist in 24 successive frames.

6.2.2 Camera Defocus Detection

The frames captured during camera defocus lack small scale details of the intended FOV causing reduced visibility. The edge details present in these frames will be less as opposed to *mosaic*. Defocusing leads to the elimination of weak edges and also reduces the sharpness (by spreading the edges) of strong edges. This in turn reduces the strength of edges. The average edge magnitude of cur_f and its overlapping region in *mosaic* are estimated for identifying camera defocus. The edge detector employed here is "Sobel". The average edge magnitudes of cur_f and the corresponding region in *mosaic* are represented as avg_edge_{cur} and avg_edge_{mosaic} respectively. Typically, $avg\_edge_{cur} \geq avg\_edge_{mosaic}$ because of the presence of foreground objects in cur_f. If foreground objects are absent, $avg\_edge_{cur} = avg\_edge_{mosaic}$ in ideal cases. This is because the edges in cur_f exists in its corresponding *mosaic* region. The presence of smooth foreground object may reduce edges in cur_f. But, this reduction is considerably less in normal cases or else the foreground should be large enough and such objects will be detected as suspicious in the partial camera occlusion detection stage (Subsection 6.2.1).

$$\frac{avg\_edge_{cur}}{avg\_edge_{mosaic}} \leq Th_{df} \tag{6.13}$$

where the threshold, Th_{df}, should be ≈ 1 and is empirically set to 0.97. If camera defocus is detected in 24 or more successive frames, then the corresponding alarm is activated. The flowchart for camera defocus detection is presented in Fig. 6.4. The outcomes of a camera defocus test are returned to the "Check Defocus"

decision box in Fig. 6.2.

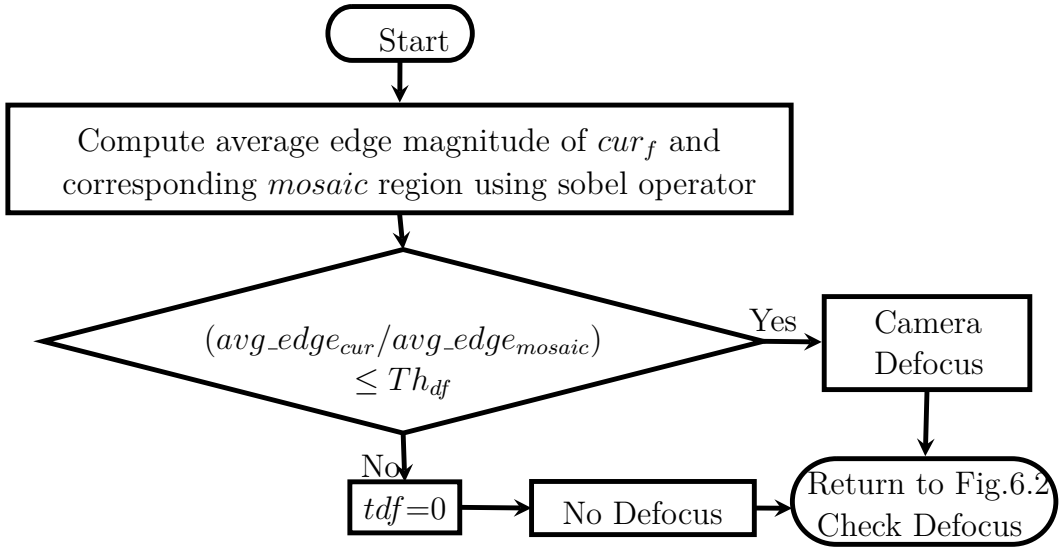


Figure 6.4: Flowchart of the proposed method for detecting camera defocus in panning surveillance systems

6.2.3 Camera Displacement Detection

In a panning surveillance system, camera displacement tampering can be performed in two ways as explained underneath.

Case 1 A surveillance camera is tilted down or up from its current position so that it is not allowed to capture specific regions of the intended FOV. As the new regions in the incoming frames concerning camera displacement is absent in the intended FOV (*mosaic*), the camera tamper detection system classifies these regions as foreground objects in foreground object extraction stage. Hence, the method developed for detecting case 2 partial camera occlusion (Subsection 6.2.1) can be used.

Case 2 The panning movement of a surveillance camera is restricted so that it cannot capture the intended FOV entirely. As a result, the number

of frames captured per pan cycle may be less as opposed to the normal working condition. In Section 6.1.1, we explained that a sign change in $R(1,3)$ indicates a border frame. We take the count of frames captured between one sign change to another and store it as frames per pan cycle (f_{count}). If f_{count} is less than the expected frame count in a pan cycle, it may be a camera displacement tampering. Tamper classification based on this single feature may contribute to false positives. Sometimes, a sign change is caused by camera noise. The incoming frame, in this case, is not a boundary of FOV. If the sign change is because of the change in camera direction, then the frames that follow current frame will also share the same sign. Therefore, finding the sign of $H(1,3)$ in those frames is useful. The sign of $H(1,3)$ obtained for cur_f is kept in c_{sign}. If sign change occurs, then the sign of $H(1,3)$ between cur_f and 2 frames that immediately follow cur_f are computed and stored as $n1_{sign}$ and $n2_{sign}$ respectively. If there is a change in the direction of movement of camera, then c_{sign}, $n1_{sign}$ and $n2_{sign}$ share the same sign.

$$(c_{sign} == n1_{sign})\&\&(c_{sign} == n2_{sign}) \tag{6.14}$$

If Eq. (6.14) does not hold, then we can discard the sign change and increment f_{count} to keep track of frames per pan cycle. If Eq. (6.14) is evaluated true, then current f_{count} is compared with tf_{count} (expected number of frames per pan cycle specific to surveillance system) for determining the anomaly.

$$(tf_{count} - Th_{dp}) \leq f_{count} \leq (tf_{count} + Th_{dp}) \tag{6.15}$$

As the panning motion and frame rate of a PTZ camera is not synchronized, we applied a loose thresholding. We kept an upper bound and lower bound for thresholding by keeping Th_{dp} number of frames on both ends. In the training stage, the count of frames per pan cycle can be recorded. The difference of the minimum and maximum frame counts obtained during training stage is assigned to Th_{dp}. If Eq. (6.15) holds, it represents a camera tamper anomaly due to displacement. An alarm is activated for prolonged camera displacement by keeping track of its occurrence (more

than 3 times). Therefore, Th_{c2} is empirically set to 3. The flowchart of the method developed for detecting camera displacement anomaly is presented in Fig. 6.5. $prev_{sign}$ and $flag_{dp}$ is initialized with 0. The validity of H estimated from $mosaic$ and cur_f have to be checked before further processing. As we skip 1 frame in each iteration, f_{count} is initialized with -1. $flag_{dp}$ is reset to 1, as it was kept for getting value in $prev_{sign}$. The sign of $H(1,3)$ is assigned to $prev_{sign}$. When the system recognizes camera movement in opposite direction, $flag_{dp}$ is reset to 0. This way it can reset f_{count} and $prev_{sign}$ for the next iteration. The outcomes of a camera displacement test are returned to the "Check Displacement" decision box in Fig. 6.2.

Perpetrators can alter (increase) the speed of movement of the camera. In such cases, the frames captured are unclear because of the fast camera motion. Frames captured per pan cycle will be less than that in normal working conditions. Therefore, the proposed system identifies this as part of camera defocus as well as displacement anomalies. The flow of control during this event is presented in Fig. 6.2. Camera tamper detection system always keeps track of frame count per pan cycle and checks camera displacement regardless of partial occlusion and defocus anomalies.

If the camera is rotating 360°, then the concept discussed in Subsection 6.1.1 for handling such cameras need to be followed for counting the frames captured per pan cycle. The rest of steps in camera displacement detection is the same.

6.3 Experimental Results and Discussion

6.3.1 Experimental Setup

The proposed method is implemented in MATLAB R2013a (8.1.0.604) on a 3.40 GHz Intel Core i7 PC. Each video frame is converted to grayscale images for background modeling and camera tamper detection. The parameter C used in foreground object extraction stage (Subsection 6.1.2) is experimentally set to 7 by trial and error method.

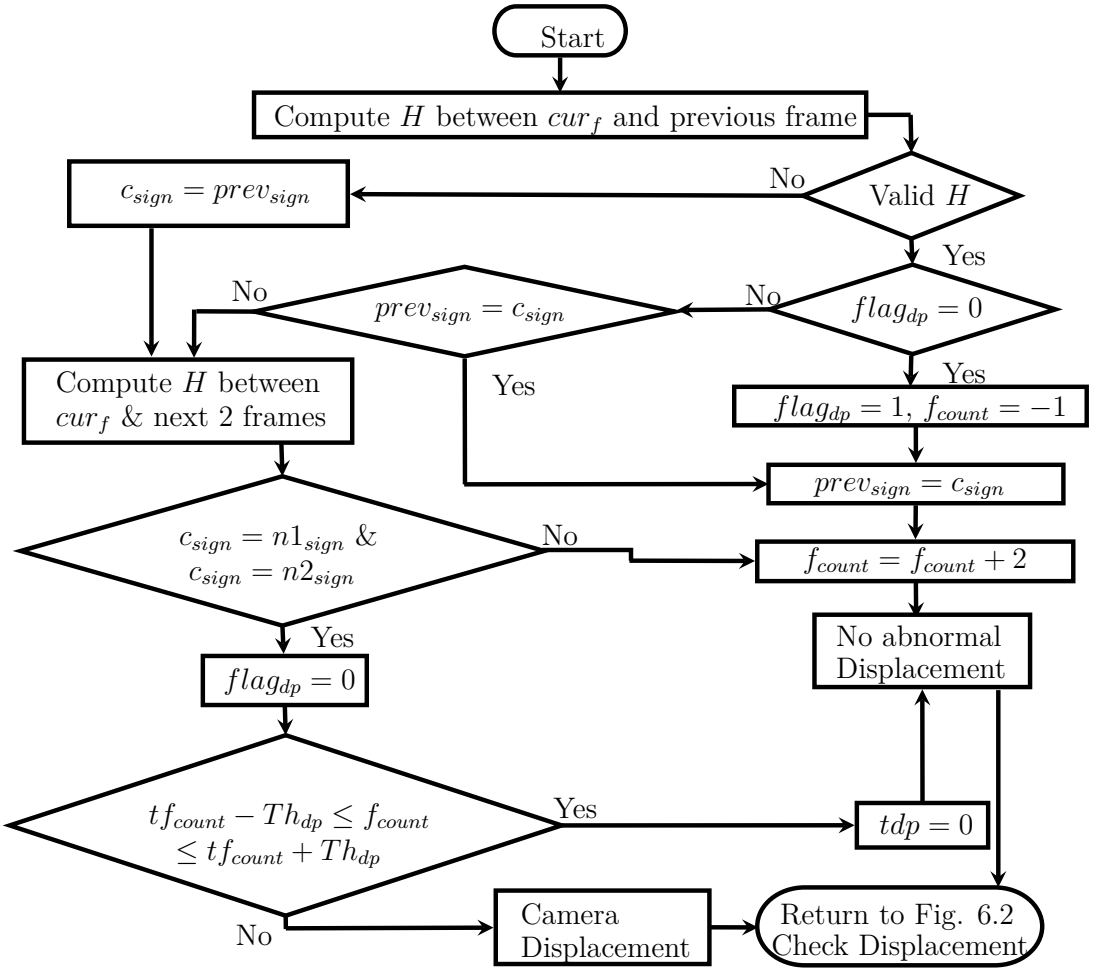


Figure 6.5: Flowchart of the proposed method for detecting Case 2 camera displacement in panning surveillance systems

6.3.2 Surveillance Camera Sabotage Dataset

For evaluating the performance of the method presented in this chapter, we conduct experiments using the panning surveillance dataset provided by Tsesmelis et al. [122]. The details regarding this dataset are discussed in Subsection 2.7.4 of Chapter 2.

Figure 6.6 and 6.7 present the frames from [122] having the visuals of camera tamper events (corresponding to the events discussed in Section 6.2) and their intended FOVs. The visual discrepancies between successive frames are consider-

ably less. Therefore, frames which are temporally separated by a frame gap of 10 are presented in Fig. 6.6 and 6.7. This makes it easy to understand the changes in the behavior of objects and tamper events. Figure 6.6 (a), the top row, displays the intended FOV frames of a surveillance camera (foyer background). Tampered frames captured during partial camera occlusion (case 1 in Subsection 6.2.1) are shown in Fig. 6.6 (b). From the frames, it can be inferred that the object is placed adjacent to the camera on it left side and it is static in the intended FOV. As the camera pans from right to left, the area occupied by this object in captured frame increases which in turn decreases the intended FOV. Tampered frames captured during partial camera occlusion (case 2 in Subsection 6.2.1) are shown in Fig. 6.6 (c). From the frames, it can be inferred that the object is placed on the camera and the amount of occlusion in consecutive frames is the same. Figure 6.6 (d) shows the frames recorded during full occlusion. The visuals of intended FOV is lost entirely. The object can be placed on camera or adjacent to it, provided it is large enough to occlude the entire FOV. Tampered frames captured during camera defocus anomaly is shown in Fig. 6.7 (b). By comparing the frames from Fig. 6.7 (a) and (b), we can deduce that camera defocus results reduced visibility. Tampered frames captured during camera displacement (case 1 in Subsection 6.2.3) are given in Fig. 6.7 (c). The intended FOV is changed by pushing the camera upwards physically. The visuals (activities) of the lower regions of the intended FOV is missing in recorded frames.

To the best of our knowledge, in the literature, the work by Tsesmelis et al. [122] is the single available work that discussed methods for addressing camera tamper detection in panning surveillance systems. Hence, the dataset provided by Saglam and Temizel [91] is used for evaluating the performance of the proposed methods with others in static surveillance systems. The details of this dataset is discussed in Subsection 2.7.3 of Chapter 2.

6.3.3 Performance Evaluation

The performance is evaluated in terms of TP, FN, TN and FP. Here, the description of TP, FN, TN and FP are as follows when camera tampering is considered as the positive class:

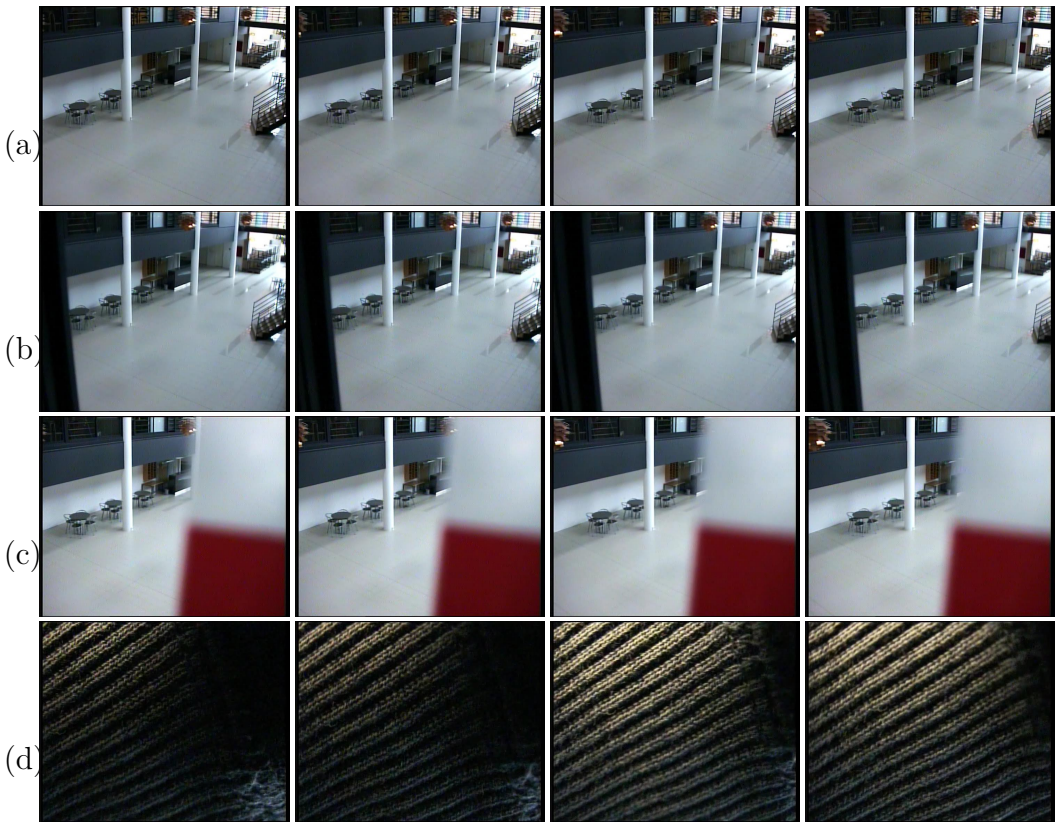


Figure 6.6: Sample frames from the dataset in [122] showing visuals of camera tamper events and their intended FOVs. (a) Intended FOV frames of a panning surveillance system. (b) Scene recorded during Case 1 partial camera occlusion. (c) Scene recorded during Case 2 partial camera occlusion. (d) Scene recorded during full camera occlusion.

- **TP** - Alarm triggered for an actual camera tamper event.
- **TN** - Camera non-tamper events categorized as non-tamper.
- **FP** - Non-tamper event wrongly categorized as tamper event.
- **FN** - Camera tamper events categorized as non-tamper.

The threshold values of the proposed system have to be tuned for better performance. To determine the threshold values, we took 60 frames from each non-tamper videos (totaling 240 frames). 60 frames each are chosen from 6 partial

Figure 6.7: Sample frames from the dataset in [122] showing visuals of camera tamper events and their intended FOVs. (a) Intended FOV frames of a panning surveillance system. (b) Scene recorded during camera defocus. (c) Scene recorded during Case 1 camera displacement.

camera occlusion tamper events where the proximity of the suspicious object is close to the camera. Thus, 360 tampered frames adhering to Case 1 partial camera occlusion discussed in Section 6.2.1 are used. The changes in FPR and FNR corresponding to different values of Th_{o1} is presented in Fig. 6.8. FNR and FPR are estimated using Eq. 3.14 and Eq. 3.15 respectively (presented in Sub-section 3.3.2 of Chapter 3). FPR is increasing after 0.98, and the FNR at that point is also better. We took 60 frames each from 3 partial camera occlusion tamper events where the suspicious object is set on the camera itself. Thus, 180 tampered frames adhering to Case 2 partial camera occlusion presented in Subsection 6.2.1 are used. The changes in FPR and FNR with respect to various values for Th_{o2} is presented in Fig. 6.9. FPR is increasing after 0.97 and the FNR at 0.96 is satisfying. We took 60 frames each from 6 camera defocus tamper events, totaling to 360 tampered frames. The changes in FPR and FNR with respect to the different values for Th_{df} is given in Fig. 6.10. FPR is increasing

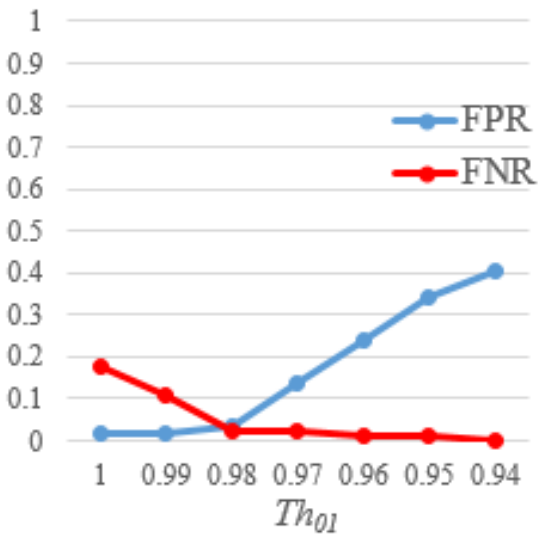


Figure 6.8: FNR and FPR variations on different threshold values for Th_{o1}.

after 0.97 and the FNR is ample. The threshold values for which almost equal error rate are obtained are selected for evaluating the performance of the proposed system on the whole dataset. From the graphs in Fig. 6.8, 6.9, and 6.10, it can be inferred that 0.98, 0.96 and 0.97 are suitable for Th_{o1}, Th_{o2} and Th_{df} respectively. FPR and FNR are almost intersecting at these values which implies equal error rate.

The performance comparison of the proposed scheme with that in [122] on their dataset is given in Table 6.1. True negatives are absent for tamper events. Hence, it is represented as 'x' in Table 6.1.

FP in tamper events indicates those set of non-tampered frames which are wrongly classified as tamper events because of an extreme change in lighting conditions. The false alarms (FP) of the proposed scheme is low compared to that of [122], without compromising its detection accuracy on camera tamper events. Tsesmelis et al. [122] kept frames separated by half frame-width distance as reference frames for determining camera tamper events in newer frames (discussed in Section 6.1). The reduction of SURF key points in newer frames compared to the stored background frame which is most similar to the newer frame is used as a feature for detecting camera defocus and occlusion. The frames coming

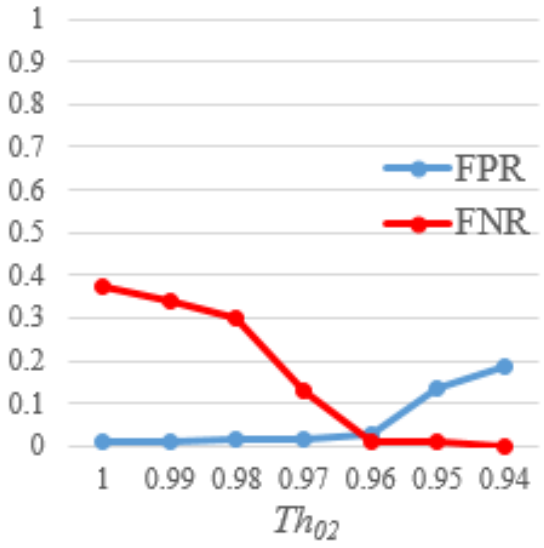


Figure 6.9: FNR and FPR variations on different threshold values for Th_{o2}.

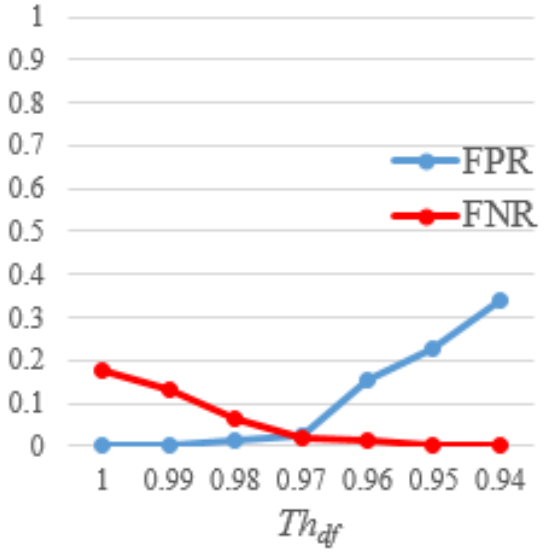


Figure 6.10: FNR and FPR variations on different threshold values for Th_{df}.

Table 6.1: Performance comparison of proposed method with Tsesmelis et al. [122] on their dataset

Type	**Videos**	**Different Approaches**	**Results**			
			TP	**FN**	**FP**	**TN**
No tamper	4	Tsesmelis et al.[122]	0	0	4	0
		Proposed method	**0**	**0**	**1**	**3**
Defocus	31	Tsesmelis et al.[122]	31	0	7	x
		Proposed method	**31**	**0**	**3**	**x**
Occlusion	14	Tsesmelis et al.[122]	14	0	6	x
		Proposed method	**14**	**0**	**2**	**x**
Displacement	31	Tsesmelis et al.[122]	31	0	10	x
		Proposed method	**31**	**0**	**4**	**x**

in between half frame-width distance may not get an exactly matching background frame. The weather and illumination changes along with the variations introduced by auto functionalities of the camera will adversely affect key point distribution. Unlike [122], the proposed method analyzes the suspicious object for its static characteristics. Then, the persistence of event is also regarded as a feature for camera tampering detection. This helps in reducing the false alarms. If the false alarm rate is high, the operator in charge may neglect system alarms expecting wrong ones. Therefore, the proposed method is more accurate and better suited for real-world implementation than [122]. To improve the robustness of the system, it is better to conduct background modeling at regular intervals.

The performance comparison of the proposed method with existing methods in [91], [122] and [100] on static surveillance video dataset from [91] are given in Table 6.2. Saglam and Temizel [91] have compared their work with existing methods in [30], [4] and [87].

The entire tamper events of [91] are unavailable. Hence, a conclusion on the performance of proposed method cannot be made. We can say that our method is performing equal or better (with less FP) compared to the existing ones.

Table 6.2: Performance comparison of the proposed method with existing methods on static surveillance dataset from [91]

Type	**Different Approaches**	**Tamper Events**	**Results**		
			TP	**FN**	**FP**
Defocus	Gil-Jiménez et al. [30]	35	22	13	1
	Aksay et al. [4]	35	28	7	5
	Ribnick et al. [87]	35	21	14	0
	Saglam & Temizel [91]	35	29	6	0
	Tsesmelis et al. [122]	13	11	2	5
	Sitara & Mehtre [100]	13	12	1	0
	Proposed method	**13**	**12**	**1**	**0**
Occlusion	Gil-Jiménez et al. [30]	40	35	5	14
	Aksay et al. [4]	40	33	7	2
	Ribnick et al. [87]	40	37	3	1
	Saglam & Temizel [91]	40	38	2	0
	Tsesmelis et al. [122]	17	17	0	2
	Sitara & Mehtre [100]	17	15	2	0
	Proposed method	**17**	**17**	**0**	**0**
Displacement	Gil-Jiménez et al. [30]	12	8	4	3
	Ribnick et al. [87]	12	7	5	0
	Saglam & Temizel [91]	12	11	1	0
	Tsesmelis et al. [122]	12	12	0	3
	Sitara & Mehtre [100]	12	11	1	0
	Proposed method	**12**	**11**	**1**	**0**

6.4 Summary

In this chapter, we presented an automatic method for camera tamper detection in panning surveillance systems. It can identify camera tampering caused by occlusion, defocus and displacement. Camera occlusion is identified by employing features like area occupied, location and static nature of foreground objects. The loss of edge information is used for camera defocus detection. Camera displacement is identified by using the number of frames recorded per pan cycle and the FOV captured.

The effectiveness of the proposed method for camera tampering detection on panning surveillance system is tested on a public dataset by comparing it with an existing method. The results show that the proposed method is better than the existing method with less false alarms. It also has equal performance regarding true positives. The performance of the proposed scheme is also assessed on static surveillance systems by comparing it with six other existing methods on a public dataset. The outcomes obtained are encouraging.

As the method exploits edge information, it may not be useful in situations where the background model has no edges or considerably less edge information. To reduce the false alarm rate, we are planning to incorporate depth information in object detection phase. These two issues are considered as future research directions of this work.

Chapter 7

Conclusions and Future Work

The detection of video and camera tampering is an inevitable part of video forensics. Camera tampering detection helps in crime prevention by timely operator intervention. It also prevents a perpetrator from physically attacking the camera during criminal activities so that the visuals are properly recorded. This helps an investigator by providing the exact visuals of crime. Video tampering detection is very much required for validating the authenticity of videos for submitting the video as digital evidence.

7.1 Conclusions

Video tampering detection

- Among various inter-frame forgeries, frame shuffling is one of the important tampering which was not addressed in the literature.
- Videos containing frames captured during camera zooming added false positives in previous works. We fixed this issue by introducing a method for zooming detection and consolidated the same with video tampering detection.
- The methods developed for inter-frame video tampering detection are tested on a custom built dataset consisting of 23586 videos in VBR and CBR coding schemes that comprise pristine and tampered videos with various

types of inter-frame alterations. The results show that our method performs better compared to the existing techniques.

- Our method can distinguish the type of inter-frame tampering applied and is successful in identifying the temporal locations of tampering. In frame shuffling and duplication forgeries, the original frames used for performing these forgeries are also discovered.
- Our method is capable of detecting inter-frame tampering in fixed as well as adaptive GOP structure videos.
- Synthetic zooming detection, hitherto an unexplored area in video forgery detection is also addressed.
- The method developed for differentiating synthetic zooming from optical zooming is tested on a custom built dataset consisting of 3200 videos with CBR and VBR coding. Experimental results show the effectiveness of this method on various noise types and compression settings.

Camera tampering detection

- A simple and efficient method for detecting camera tamper events in static surveillance systems is proposed. It is based on background modeling, edge components, foreground object's area, and its relative position in the previous, current, and future frames.
- Experimental results on video sequences from public datasets containing camera tamper events and non-tamper events demonstrate the effectiveness of our method. The results show that compared to existing methods, our method has equal performance in detecting the true camera tamper events and has reduced false positives.
- A method for camera tampering detection in panning surveillance systems is presented. It is tested using public datasets for camera tampering detection on panning and static surveillance systems.

- It shows that compared to existing methods, the proposed method has reduced false positives and has equal performance in detecting the true camera tamper events. Hence, the proposed method can be implemented with high precision and low false alarm rate.

7.2 Future Research Directions

The following issues are considered for future research:

- Exploring how the proposed methods can be utilized for the detection of multiple tampering present in a single video.
- Frame shuffling detection with single tamper point (shuffled frames placed anywhere other than the beginning or ending of video).
- Exploration of anti-forensic methods that can be applied over the proposed video tamper detection system. The methods for countering the anti-forensic aspects can be considered.
- As the methods proposed for camera tamper detection use edge information, it may not be effective for situations where the background model has no edges or less edge information. This need to be further investigated.
- Videos subjected to copy-move tampering using affine transformation have to be detected and regions of tampering have to be found.

List of Publications

- **International Peer-Reviewed Journals**

1. **Sitara K.** and B. M. Mehtre, "Detection of Inter-Frame Forgeries in Digital Videos", *Forensic Science International*, Volume 289, Pages 186-206, Available online 26 May 2018, ISSN 0379-0738. doi:10.1016/j.forsciint.2018.04.056
   Publisher: Elsevier **Impact Factor:** 1.989
   Indexing: Science Citation Index, Scopus
2. **Sitara K.** and B. M. Mehtre, "Automated Camera Sabotage Detection for Enhancing Video Surveillance Systems", *Multimedia Tools and Applications*, Jun 2018, ISSN 1573-7721. doi:10.1007/s11042-018-6165-4.
   Publisher: Springer **Impact Factor:** 1.53
   Indexing: Science Citation Index Expanded, Scopus, DBLP
3. **Sitara K.** and B. M. Mehtre, "Digital video tampering detection: An overview of passive techniques", *Digital Investigation*, Volume 18, Pages 8-22, September 2016, doi:10.1016/j.diin.2016.06.003.
   Publisher: Elsevier **Impact Factor:** 1.774
   Indexing: Science Citation Index Expanded, Scopus, DBLP
4. **Sitara K.** and B. M. Mehtre, "Differentiating Synthetic and Natural Zooming for Passive Video Forgery Detection: An Anti-Forensic Perspective", *IEEE Transactions on Information Forensics and Security*, April 2018. (Under Review)
   Publisher: IEEE **Impact Factor:** 4.332
   Indexing: Science Citation Index, Scopus, DBLP

- **Conference Proceedings**

5. **Sitara K.**, and B. M. Mehtre, "A Comprehensive Approach for Exposing Inter-Frame Video Forgeries", *2017 IEEE* 13^{th} *International Colloquium on Signal Processing & Its Applications (CSPA)*, Batu Feringgi, Malaysia, 2017, pp. 73-78. doi:10.1109/CSPA.2017.8064927
   Indexing: Scopus

6. **Sitara K.**, and B. M. Mehtre, "Real-Time Automatic Camera Sabotage Detection for Surveillance Systems", *Second International Symposium on Signal Processing and Intelligent Recognition Systems (SIRS-2015)*, Trivandrum, Dec 16-19, 2015, pp. 75-84, Springer International Publishing, 2016. doi:10.1007/978-3-319-28658-7_7.
   Indexing: Scopus, DBLP

- **Book Chapter (*In Press*)**

7. **Sitara K.** and B. M. Mehtre."Anti-Forensics for Image & Video Tampering: A Review", *Cryptographic and Information Security Approaches for Images and Videos*, Editor: Dr.S.Ramakrishnan, CRC Press, Taylor and Francis Group, [*In Press*]

References

[1] A. J. Abbasi and A. Behrad. Malicious inter-frame video tampering detection in MPEG videos using time and spatial domain analysis of quantization effects. *Multimedia Tools and Applications*, 76(20):20691–20717, Oct 2017. ISSN 1573-7721. doi:10.1007/s11042-016-4004-z.

[2] E. E. Abdallah, A. Ben Hamza, and P. Bhattacharya. Video watermarking using wavelet transform and tensor algebra. *Signal, Image and Video Processing*, 4(2):233–245, Jun 2010. ISSN 1863-1711. doi:10.1007/s11760-009-0114-7.

[3] S. A. Ahmed, D. P. Dogra, S. Kar, B. G. Kim, P. Hill, and H. Bhaskar. Localization of region of interest in surveillance scene. *Multimedia Tools and Applications*, 76(11):13651–13680, Jun 2017. ISSN 1573-7721. doi:10.1007/s11042-016-3762-y.

[4] A. Aksay, A. Temizel, and A. Enis Cetin. Camera tamper detection using wavelet analysis for video surveillance. In *IEEE Conference on Advanced Video and Signal Based Surveillance, AVSS 2007.*, pages 558–562, Sept 2007. doi:10.1109/AVSS.2007.4425371.

[5] E. Ardizzone and G. Mazzola. A tool to support the creation of datasets of tampered videos. In *18th International Conference on Image Analysis and Processing - ICIAP 2015*, pages 665–675. Springer International Publishing, 2015. ISBN 978-3-319-23234-8. doi:10.1007/978-3-319-23234-8_61.

[6] M. P. J. Ashby. The value of CCTV surveillance cameras as an investigative tool: An empirical analysis. *European Journal on Criminal Policy and*

Research, 23(3):441–459, Sep 2017. ISSN 1572-9869. doi:10.1007/s10610-017-9341-6.

[7] P. Bestagini, A. Allam, S. Milani, M. Tagliasacchi, and S. Tubaro. Video codec identification. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2257–2260, March 2012. doi:10.1109/ICASSP.2012.6288363.

[8] P. Bestagini, S. Battaglia, S. Milani, M. Tagliasacchi, and S. Tubaro. Detection of temporal interpolation in video sequences. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3033–3037. IEEE, 2013.

[9] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro. Local tampering detection in video sequences. In *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, pages 488–493. IEEE, 2013.

[10] A. Bidokhti and S. Ghaemmaghami. Detection of regional copy/move forgery in MPEG videos using optical flow. In *2015 International Symposium on Artificial Intelligence and Signal Processing (AISP)*, pages 13–17. IEEE, March 2015. doi:10.1109/AISP.2015.7123529.

[11] A. M. Buhari, H. C. Ling, V. M. Baskaran, and K. Wong. Low complexity watermarking scheme for scalable video coding. In *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2, May 2016. doi:10.1109/ICCE-TW.2016.7520710.

[12] J. Chao, X. Jiang, and T. Sun. A novel video inter-frame forgery model detection scheme based on optical flow consistency. In *11th International Workshop on Digital Forensics and Watermaking, IWDW 2012, Shanghai, China, October 31 – November 3, 2012*, pages 267–281. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40099-5. doi:10.1007/978-3-642-40099-5_22.

[13] S. Chen, S. Tan, B. Li, and J. Huang. Automatic detection of object-based forgery in advanced video. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(11):2138–2151, Nov 2016. ISSN 1051-8215. doi:10.1109/TCSVT.2015.2473436.

[14] W. Chen and Y. Q. Shi. Detection of double MPEG compression based on first digit statistics. In *Digital Watermarking*, pages 16–30. Springer-Verlag, 2009. ISBN 978-3-642-04437-3. doi:10.1007/978-3-642-04438-0_2.

[15] G. Chetty, M. Biswas, and R. Singh. Digital video tamper detection based on multimodal fusion of residue features. In *2010 4th International Conference on Network and System Security (NSS)*, pages 606–613. IEEE, 2010.

[16] P. M. Comiskey, A. L. Yarin, and D. Attinger. High-speed video analysis of forward and backward spattered blood droplets. *Forensic Science International*, 276(Supplement C):134 – 141, 2017. ISSN 0379-0738. doi:https://doi.org/10.1016/j.forsciint.2017.04.016.

[17] D. Cozzolino, G. Poggi, and L. Verdoliva. Copy-move forgery detection based on patchmatch. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 5312–5316, Oct 2014. doi:10.1109/ICIP.2014.7026075.

[18] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space. In *2007 IEEE International Conference on Image Processing*, volume 1, pages I – 313–I – 316, Sept 2007. doi:10.1109/ICIP.2007.4378954.

[19] L. D'Amiano, D. Cozzolino, G. Poggi, and L. Verdoliva. Video forgery detection and localization based on 3D patchmatch. In *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–6, June 2015. doi:10.1109/ICMEW.2015.7169805.

[20] Q. Dong, G. Yang, and N. Zhu. A MCEA based passive forensics scheme for detecting frame-based video tampering. *Digital Investigation*, 9(2):151–159, 2012.

[21] D. Ellwart, P. Szczuko, and A. Czyżewski. Camera sabotage detection for surveillance systems. In *Proceedings of the 2011 International Conference on Security and Intelligent Information Systems*, SIIS'11, pages 45–53, Berlin, Heidelberg, 2012. Springer-Verlag. doi:10.1007/978-3-642-25261-7_4.

[22] M. Fallahpour, S. Shirmohammadi, M. Semsarzadeh, and J. Zhao. Tampering detection in compressed digital video using watermarking. *IEEE Transactions on Instrumentation and Measurement*, 63(5):1057–1072, May 2014. ISSN 0018-9456. doi:10.1109/TIM.2014.2299371.

[23] O. S. Faragallah. Efficient video watermarking based on singular value decomposition in the discrete wavelet transform domain. *AEU - International Journal of Electronics and Communications*, 67(3):189 – 196, 2013. ISSN 1434-8411. doi:http://doi.org/10.1016/j.aeue.2012.07.010.

[24] C. Feng, Z. Xu, W. Zhang, and Y. Xu. Automatic location of frame deletion point for digital video forensics. In *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, pages 171–179. ACM, 2014.

[25] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter)*, pages 1–6. IEEE, 2009.

[26] J. Ferryman and D. Tweed. An overview of the pets 2007 dataset. In *Proceeding Tenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, PETS*, 2007.

[27] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi:10.1145/358669.358692.

[28] J. Fitzsimons and K. Dawson-Howe. Abandoned, removed and moved object classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 30(01):1655002, 2016. doi:10.1142/S0218001416550028.

[29] A. C. Gallagher. Detection of linear and cubic interpolation in JPEG compressed images. In *The 2nd Canadian Conference on Computer and Robot Vision (CRV'05)*, pages 65–72. IEEE, May 2005. doi:10.1109/CRV.2005.33.

[30] P. Gil-Jiménez, R. López-Sastre, P. Siegmann, J. Acevedo-Rodríguez, and S. Maldonado-Bascón. Automatic control of video surveillance camera sabotage. In *Proceedings of the 2nd International Work-conference on Nature Inspired Problem-Solving Methods in Knowledge Engineering: Interplay Between Natural and Artificial Computation, Part II*, IWINAC '07, pages 222–231, Berlin, Heidelberg, 2007. Springer-Verlag. doi:10.1007/978-3-540-73055-2_24.

[31] A. Gironi, M. Fontani, T. Bianchi, A. Piva, and M. Barni. A video forensic technique for detecting frame deletion and insertion. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6226–6230. IEEE, 2014.

[32] J. Goodwin and G. Chetty. Blind video tamper detection based on fusion of source features. In *2011 International Conference on Digital Image Computing Techniques and Applications (DICTA)*, pages 608–613. IEEE, 2011.

[33] I. Grant. Particle image velocimetry: a review. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 211(1):55–76, 1997.

[34] A. Gupta, S. Gupta, and A. Mehra. Video authentication in digital forensic. In *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pages 659–663. IEEE, Feb 2015. doi:10.1109/ABLAZE.2015.7154945.

[35] K. K. Hati, P. K. Sa, and B. Majhi. Intensity range based background subtraction for effective object detection. *IEEE Signal Processing Letters*, 20 (8):759–762, Aug 2013. ISSN 1070-9908. doi:10.1109/LSP.2013.2263800.

[36] P. He, X. Jiang, T. Sun, and S. Wang. Double compression detection based on local motion vector field analysis in static-background videos. *Journal of Visual Communication and Image Representation*, 35:55 – 66, 2016. ISSN 1047-3203. doi:http://dx.doi.org/10.1016/j.jvcir.2015.11.014.

[37] C. C. Hsu, T. Y. Hung, C. W. Lin, and C. T. Hsu. Video forgery detection using correlation of noise residue. In *2008 IEEE 10th Workshop on Multimedia Signal Processing*, pages 170–174. IEEE, 2008.

[38] C. C. Huang, Y. Zhang, and V. L. L. Thing. Inter-frame video forgery detection based on multi-level subtraction approach for realistic video forensic applications. In *2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)*, pages 20–24, Aug 2017. doi:10.1109/SIPROCESS.2017.8124498.

[39] D. Y. Huang, C. H. Chen, T. Y. Chen, W. C. Hu, and B. C. Chen. Rapid detection of camera tampering and abnormal disturbance for video surveillance system. *Journal of Visual Communication and Image Representation*, 25(8):1865–1877, 2014. doi:http://dx.doi.org/10.1016/j.jvcir.2014.09.007.

[40] Z. Huang, F. Huang, and J. Huang. Detection of double compression with the same bit rate in MPEG-2 videos. In *2014 IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP)*, pages 306–309. IEEE, 2014.

[41] D. K. Hyun, M. J. Lee, S. J. Ryu, H. Y. Lee, and H. K. Lee. Forgery detection for surveillance video. In *The Era of Interactive Media*, pages 25–36. Springer New York, New York, NY, 2013. ISBN 978-1-4614-3501-3. doi:10.1007/978-1-4614-3501-3_3.

[42] D. K. Hyun, S. J. Ryu, H. Y. Lee, and H. K. Lee. Detection of upscale-crop and partial manipulation in surveillance video based on sensor pattern noise. *Sensors*, 13(9):12605–12631, 2013.

[43] i-LIDS dataset for AVSS 2007. http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007_d.html. Date last accessed 8 July 2015).

[44] B. Iglewicz and D. C. Hoaglin. *How to detect and handle outliers*, volume 16. Asq Press, 1993.

[45] M. Jerian, S. Paolino, F. Cervelli, S. Carrato, A. Mattei, and L. Garofano. A forensic image processing environment for investigation of surveillance video. *Forensic Science International*, 167(2):207 – 212, 2007. ISSN 0379-0738. Selected Articles of the 4th European Academy of Forensic Science Conference (EAFS2006) June 13-16, 2006 Helsinki, Finland. doi:https://doi.org/10.1016/j.forsciint.2006.06.048.

[46] S. Jia, Z. Xu, H. Wang, C. Feng, and T. Wang. Coarse-to-fine copy-move forgery detection for video forensics. *IEEE Access*, 6:25323–25335, 2018. doi:10.1109/ACCESS.2018.2819624.

[47] X. Jiang, W. Wang, T. Sun, Y. Q. Shi, and S. Wang. Detection of double compression in MPEG-4 videos based on markov statistics. *IEEE Signal Processing Letters*, 20(5):447–450, 2013.

[48] V. Joshi and S. Jain. Tampering detection in digital video - a review of temporal fingerprints based techniques. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1121–1124. IEEE, March 2015.

[49] X. Kang, J. Liu, H. Liu, and Z. J. Wang. Forensics and counter anti-forensics of video inter-frame forgery. *Multimedia Tools and Applications*, 75(21):13833–13853, Nov 2016. ISSN 1573-7721. doi:10.1007/s11042-015-2762-7.

[50] S. Kingra, N. Aggarwal, and R. D. Singh. Inter-frame forgery detection in H.264 videos using motion and brightness gradients. *Multimedia Tools and Applications*, 76(24):25767–25786, Dec 2017. ISSN 1573-7721. doi:10.1007/s11042-017-4762-2.

[51] M. Kirchner. Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue. In *Proceedings of the 10th ACM Workshop on Multimedia and Security*, MM & Sec '08, pages 11–20, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-058-6. doi:10.1145/1411328.1411333.

[52] M. Kirchner. Linear row and column predictors for the analysis of resized images. In *Proceedings of the 12th ACM Workshop on Multimedia and Security*, MM & Sec '10, pages 13–18, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0286-9. doi:10.1145/1854229.1854234.

[53] M. Kirchner and T. Gloe. On resampling detection in re-compressed images. In *2009 First IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 21–25, Dec 2009. doi:10.1109/WIFS.2009.5386489.

[54] M. Kobayashi, T. Okabe, and Y. Sato. Detecting forgery from static-scene video based on inconsistency in noise level functions. *IEEE Transactions on Information Forensics and Security*, 5(4):883–892, 2010.

[55] T. Kryjak, M. Komorkiewicz, and M. Gorgon. FPGA implementation of camera tamper detection in real-time. In *Conference on Design and Architectures for Signal and Image Processing (DASIP), 2012*, pages 1–8. IEEE, Oct 2012.

[56] D. Labartino, T. Bianchi, A. De Rosa, M. Fontani, D. Vazquez-Padin, A. Piva, and M. Barni. Localization of forgeries in MPEG-2 video through GOP size and DQ analysis. In *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, pages 494–499. IEEE, 2013.

[57] G. Lavee, L. Khan, and B. Thuraisingham. A framework for a video analysis tool for suspicious event detection. *Multimedia Tools and Applications*, 35(1): 109–123, Oct 2007. ISSN 1573-7721. doi:10.1007/s11042-007-0117-8.

[58] J. Lee, E. D. Lee, H. O. Tark, J. W. Hwang, and D. Y. Yoon. Efficient height measurement method of surveillance camera image. *Forensic Science International*, 177(1):17 – 23, 2008. ISSN 0379-0738. doi:https://doi.org/10.1016/j.forsciint.2007.10.008.

[59] C. Li, Q. Ma, L. Xiao, M. Li, and A. Zhang. Image splicing detection based on markov features in QDCT domain. *Neurocomputing*, 228:29 – 36, 2017. ISSN 0925-2312. doi:http://doi.org/10.1016/j.neucom.2016.04.068. Advanced Intelligent Computing: Theory and Applications.

[60] H. Li and S. Forchhammer. MPEG2 video parameter and no reference PSNR estimation. In *Picture Coding Symposium, PCS 2009*, pages 1–4. IEEE, 2009.

[61] L. Li, X. Wang, W. Zhang, G. Yang, and G. Hu. Detecting removed object from video with stationary background. In *Proceedings of the 11th International Conference on Digital Forensics and Watermaking*, IWDW'12, pages 242–252, Berlin, Heidelberg, 2013. Springer-Verlag. ISBN 978-3-642-40098-8. doi:10.1007/978-3-642-40099-5_20.

[62] N. Li, X. Wu, H. Guo, D. Xu, Y. Ou, and Y. L. Chen. Anomaly detection in video surveillance via gaussian process. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(06):1555011, 2015. doi:10.1142/S0218001415550113.

[63] C. Lin and J. Tsay. A passive approach for effective detection and localization of region-level video forgery with spatio-temporal coherence analysis. *Digital Investigation*, 11(2):120–140, 2014.

[64] D. T. Lin and C. H. Wu. Real-time active tampering detection of surveillance camera and implementation on digital signal processor. In *Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2012*, pages 383–386. IEEE, July 2012. doi:10.1109/IIH-MSP.2012.99.

[65] G. S. Lin, J. F Chang, and C. H. Chuang. Detecting frame duplication based on spatial and temporal analyses. In *2011 6th International Conference on Computer Science & Education (ICCSE)*, pages 1396–1399. IEEE, 2011.

[66] H. Liu, S. Li, and S. Bian. Detecting frame deletion in H.264 video. In *International Conference on Information Security Practice and Experience*, pages 262–270. Springer, 2014.

[67] Y. Liu and T. Huang. Exposing video inter-frame forgery by zernike opponent chromaticity moments and coarseness analysis. *Multimedia Systems*, 23(2): 223–238, Mar 2017. ISSN 1432-1882. doi:10.1007/s00530-015-0478-1.

[68] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi:10.1023/B:VISI.0000029664.99615.94.

[69] W. Luo, M. Wu, and J. Huang. MPEG recompression detection based on block artifacts. In *Electronic Imaging 2008*, pages 68190X–68190X. International Society for Optics and Photonics, 2008.

[70] B. Mahdian and S. Saic. On periodic properties of interpolation and their application to image authentication. In *Third International Symposium on Information Assurance and Security*, pages 439–446. IEEE, Aug 2007. doi:10.1109/IAS.2007.49.

[71] B. Mahdian and S. Saic. Blind authentication using periodic properties of interpolation. *IEEE Transactions on Information Forensics and Security*, 3 (3):529–538, Sept 2008. ISSN 1556-6013. doi:10.1109/TIFS.2004.924603.

[72] B. Mahdian and S. Saic. Detection of resampling supplemented with noise inconsistencies analysis for image forensics. In *2008 International Conference on Computational Sciences and Its Applications*, pages 546–556. IEEE, June 2008. doi:10.1109/ICCSA.2008.34.

[73] V. T. Manu and B. M. Mehtre. Copy-move tampering detection using affine transformation property preservation on clustered keypoints. *Signal, Image and Video Processing*, 12(3):549–556, Mar 2018. ISSN 1863-1711. doi:10.1007/s11760-017-1191-7.

[74] S. Milani, P. Bestagini, M. Tagliasacchi, and S. Tubaro. Multiple compression detection for video sequences. In *2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP)*, pages 112–117. IEEE, 2012.

[75] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro. An overview on video forensics. *APSIPA Transactions on Signal and Information Processing*, 1:e2, 2012.

[76] P. K. Mishra and I. Hooda. Robust adaptive watermarking in video for protecting intellectual properties. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 3128–3131. IEEE, March 2016.

[77] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *International Conference on Computer Vision Theory and Application (VISSAPP'09)*, pages 331–340, 2009.

[78] R. C. Pandey, S. K. Singh, and K. K. Shukla. Passive copy-move forgery detection in videos. In *2014 International Conference on Computer and Communication Technology (ICCCT)*, pages 301–306. IEEE, 2014.

[79] PETS. Performance Evaluation of Tracking and Surveillance Workshop, 2001, http://pets2001.visualsurveillance.org. (Date last accessed 14-Dec-2016).

[80] A. C. Popescu and H. Farid. Exposing digital forgeries by detecting traces of resampling. *IEEE Transactions on Signal Processing*, 53(2):758–767, Feb 2005. ISSN 1053-587X. doi:10.1109/TSP.2004.839932.

[81] S. Prasad and K. R. Ramakrishnan. On resampling detection and its application to detect image tampering. In *2006 IEEE International Conference on Multimedia and Expo*, pages 1325–1328, July 2006. doi:10.1109/ICME.2006.262783.

[82] C. M. Pun, B. Liu, and X. C. Yuan. Multi-scale noise estimation for image splicing forgery detection. *Journal of Visual Communication and Image Representation*, 38:195 – 206, 2016. ISSN 1047-3203. doi:http://dx.doi.org/10.1016/j.jvcir.2016.03.005.

[83] G. Qadir, S. Yahaya, and A. T. S. Ho. Surrey university library for forensic analysis (SULFA) of video content. In *IET Conference on Image Processing (IPR 2012)*, pages 1–6, July 2012. doi:10.1049/cp.2012.0422.

[84] N. Ramstrand, S. Ramstrand, P. Brolund, K. Norell, and P. Bergström. Relative effects of posture and activity on human height estimation from surveillance footage. *Forensic Science International*, 212(1):27 – 31, 2011. ISSN 0379-0738. doi:https://doi.org/10.1016/j.forsciint.2011.05.002.

[85] P. Rasti, S. Samiei, M. Agoyi, S. Escalera, and G. Anbarjafari. Robust non-blind color video watermarking using QR decomposition and entropy analysis. *Journal of Visual Communication and Image Representation*, 38:838 – 847, 2016. ISSN 1047-3203. doi:http://dx.doi.org/10.1016/j.jvcir.2016.05.001.

[86] H. Ravi, A. V. Subramanyam, G. Gupta, and B. A. Kumar. Compression noise based video forgery detection. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 5352–5356. IEEE, 2014.

[87] E. Ribnick, S. Atev, O. Masoud, N. Papanikolopoulos, and R. Voyles. Real-time detection of camera tampering. In *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance*, AVSS '06, pages 10–15, Washington, DC, USA, 2006. IEEE Computer Society. doi:10.1109/AVSS.2006.94.

[88] I. E. Richardson. *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2003.

[89] A. Rocha, W. Scheirer, T. Boult, and S. Goldenstein. Vision of the unseen: Current trends and challenges in digital image and video forensics. *ACM Computing Surveys (CSUR)*, 43(4):26, 2011.

[90] P. Russo, E. Gualdi-Russo, A. Pellegrinelli, J. Balboni, and A. Furini. A new approach to obtain metric data from video surveillance: Preliminary evaluation of a low-cost stereo-photogrammetric system. *Forensic Science International*, 271(Supplement C):59 – 67, 2017. ISSN 0379-0738. doi:https://doi.org/10.1016/j.forsciint.2016.12.023.

[91] A. Saglam and A. Temizel. Real-time adaptive camera tamper detection for video surveillance. In *Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS '09.*, pages 430–435, Sept 2009. doi:10.1109/AVSS.2009.29.

[92] N. Sahu and A. Sur. SIFT based video watermarking resistant to temporal scaling. *Journal of Visual Communication and Image Representation*, 45:77 – 86, 2017. ISSN 1047-3203. doi:https://doi.org/10.1016/j.jvcir.2017.02.013.

[93] F. Scarano and M. L. Riethmuller. Iterative multigrid approach in PIV image processing with discrete window offset. *Experiments in Fluids*, 26(6):513–523, 1999. ISSN 1432-1114. doi:10.1007/s003480050318.

[94] T. Shanableh. Detection of frame deletion for digital video forensics. *Digital Investigation*, 10(4):350 – 360, 2013. ISSN 1742-2876. doi:https://doi.org/10.1016/j.diin.2013.10.004.

[95] C. C. Shih, S. C. Chen, C. F. Hung, K. W. Chen, S. Y. Lin, C. W. Lin, and Y. P. Hung. Real-time camera tampering detection using two-stage scene matching. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2013. doi:10.1109/ICME.2013.6607568.

[96] T. Sikora. Digital consumer electronics handbook. chapter Digital Video Coding Standards, pages 83–823. McGraw-Hill, Inc., Hightstown, NJ, USA, 1997. ISBN 0-07-034143-5. URL `http://dl.acm.org/citation.cfm?id=275869.275882`.

[97] R. D. Singh and N. Aggarwal. Detection of upscale-crop and splicing for digital video authentication. *Digital Investigation*, 21:31 – 52, 2017. ISSN 1742-2876. doi:http://dx.doi.org/10.1016/j.diin.2017.01.001.

[98] R. D. Singh and N. Aggarwal. Video content authentication techniques: a comprehensive survey. *Multimedia Systems*, 24(2):211–240, Mar 2018. ISSN 1432-1882. doi:10.1007/s00530-017-0538-9.

[99] V. K. Singh, P. Pant, and R. C. Tripathi. Detection of frame duplication type of forgery in digital video using sub-block based features. In *International Conference on Digital Forensics and Cyber Crime*, pages 29–38. Springer, 2015.

[100] K. Sitara and B. M. Mehtre. Real-time automatic camera sabotage detection for surveillance systems. In *Advances in Signal Processing and Intelligent Recognition Systems*, pages 75–84. Springer, 2016. doi:10.1007/978-3-319-28658-7_7".

[101] K. Sitara and B. M. Mehtre. Digital video tampering detection: An overview of passive techniques. *Digital Investigation*, 18:8 – 22, 2016. ISSN 1742-2876. doi:https://doi.org/10.1016/j.diin.2016.06.003.

[102] K. Sitara and B. M. Mehtre. A comprehensive approach for exposing inter-frame video forgeries. In *2017 IEEE 13th International Colloquium on Signal Processing its Applications (CSPA)*, pages 73–78, March 2017. doi:10.1109/CSPA.2017.8064927.

[103] J. Song, K. Lee, W. Y. Lee, and H. Lee. Integrity verification of the ordered data structures in manipulated video content. *Digital Investigation*, 18:1 – 7, 2016. ISSN 1742-2876. doi:http://doi.org/10.1016/j.diin.2016.06.001.

[104] M. C. Stamm and K. J. R. Liu. Anti-forensics for frame deletion/addition in MPEG video. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1876–1879, May 2011. doi:10.1109/ICASSP.2011.5946872.

[105] M. C. Stamm, W. S. Lin, and K. J. R. Liu. Temporal forensics and anti-forensics for motion compensated video. *IEEE Transactions on Information Forensics and Security*, 7(4):1315–1329, Aug 2012. ISSN 1556-6013. doi:10.1109/TIFS.2012.2205568.

[106] L. Su, T. Huang, and J. Yang. A video forgery detection algorithm based on compressive sensing. *Multimedia Tools Appl.*, 74(17):6641–6656, September 2015. ISSN 1380-7501. doi:10.1007/s11042-014-1915-4.

[107] P. C. Su, P. L. Suei, M. K. Chang, and J. Lain. Forensic and anti-forensic techniques for video shot editing in H.264/AVC. *Journal of Visual Communication and Image Representation*, 29:103 – 113, 2015. ISSN 1047-3203. doi:http://dx.doi.org/10.1016/j.jvcir.2015.02.006.

[108] Y. Su and J. Xu. Detection of double-compression in MPEG-2 videos. In *2010 2nd International Workshop on Intelligent Systems and Applications (ISA)*, pages 1–4. IEEE, 2010.

[109] Y. Su, J. Zhang, and J. Liu. Exposing digital video forgery by detecting motion-compensated edge artifact. In *International Conference on Computational Intelligence and Software Engineering, CiSE 2009*, pages 1–4. IEEE, 2009.

[110] Y. Su, W. Nie, and C. Zhang. A frame tampering detection algorithm for MPEG videos. In *2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 2, pages 461–464. IEEE, 2011.

[111] A. V. Subramanyam and S. Emmanuel. Video forgery detection using HOG features and compression properties. In *2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP)*, pages 89–94. IEEE, 2012.

[112] A. V. Subramanyam and S. Emmanuel. Pixel estimation based video forgery detection. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3038–3042. IEEE, 2013.

[113] N. Sulman, T. Sanocki, D. Goldgof, and R. Kasturi. How effective is human video surveillance performance? In *2008 19th International Conference on Pattern Recognition*, pages 1–3, Dec 2008. doi:10.1109/ICPR.2008.4761655.

[114] T. Sun, W. Wang, and X. Jiang. Exposing video forgeries by detecting MPEG double compression. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1389–1392. IEEE, 2012.

[115] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, Mar 1996. ISSN 0272-1716. doi:10.1109/38.486677.

[116] M. Tagliasacchi and S. Tubaro. Blind estimation of the QP parameter in H.264/AVC decoded video. In *2010 11th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4. IEEE, 2010.

[117] Team FFmpeg. FFmpeg. *URL http://ffmpeg.org*, 2017.

[118] Xiph.org test media. http://media.xiph.org/video/derf/. (Date last accessed 14-Dec-2016).

[119] Y. Tew, K. Wong, R. C. W. Phan, and K. N. Ngan. Multi-layer authentication scheme for HEVC video based on embedded statistics. *Journal of Visual Communication and Image Representation*, 40:502 – 515, 2016. ISSN 1047-3203. doi:http://dx.doi.org/10.1016/j.jvcir.2016.07.017.

[120] W. Thielicke and E. J. Stamhuis. Pivlab–towards user-friendly, affordable and accurate digital particle image velocimetry in MATLAB. *Journal of Open Research Software*, 2(1):e30.

[121] W. Thielicke and E. J. Stamhuis. PIVlab–time-resolved digital particle image velocimetry tool for MATLAB, version: 6. *figshare. Code*, 2015. doi:https://doi.org/10.6084/m9.figshare.1092508.v6.

[122] T. Tsesmelis, L. Christensen, P. Fihl, and T. B. Moeslund. Tamper detection for active surveillance systems. In *10th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 57–62, Aug 2013. doi:10.1109/AVSS.2013.6636616.

[123] C. L. Tung, P. L. Tung, and C. W. Kuo. Camera tamper detection using codebook model for video surveillance. In *International Conference on Machine Learning and Cybernetics (ICMLC), 2012*, volume 5, pages 1760–1763. IEEE, July 2012. doi:10.1109/ICMLC.2012.6359641.

[124] G. Valenzise, M. Tagliasacchi, and S. Tubaro. Estimating QP and motion vectors in H.264/AVC video from decoded pixels. In *Proceedings of the 2nd ACM workshop on Multimedia in forensics, security and intelligence*, pages 89–92. ACM, 2010.

[125] D. Vazquez-Padin, M. Fontani, T. Bianchi, P. Comesaña, A. Piva, and M. Barni. Detection of video double encoding with GOP size estimation. In *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 151–156. IEEE, 2012.

[126] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[127] L. Verdoliva, D. Cozzolino, and G. Poggi. A feature-based approach for image tampering detection and localization. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 149–154, Dec 2014. doi:10.1109/WIFS.2014.7084319.

[128] C. Vural and B. Baraklı. Adaptive reversible video watermarking based on motion-compensated prediction error expansion with pixel selection. *Signal, Image and Video Processing*, 10(7):1225–1232, Oct 2016. ISSN 1863-1711. doi:10.1007/s11760-016-0881-x.

[129] A. W. A. Wahab, M. A. Bagiwa, M. Y. I. Idris, S. Khan, Z. Razak, and M. R. K. Ariffin. Passive video forgery detection techniques: A survey. In *2014 10th International Conference on Information Assurance and Security (IAS)*, pages 29–34. IEEE, Nov 2014. doi:10.1109/ISIAS.2014.7064616.

[130] Q. Wang, Z. Li, Z. Zhang, and Q. Ma. Video inter-frame forgery identification based on consistency of correlation coefficients of gray values. *Journal of Computer and Communications*, 2(04):51, 2014.

[131] W. Wang and H. Farid. Exposing digital forgeries in video by detecting double MPEG compression. In *Proceedings of the 8th workshop on Multimedia and security*, pages 37–47. ACM, 2006.

[132] W. Wang and H. Farid. Exposing digital forgeries in video by detecting duplication. In *Proceedings of the 9th workshop on Multimedia & security*, pages 35–42. ACM, 2007.

[133] W. Wang and H. Farid. Exposing digital forgeries in interlaced and deinterlaced video. *IEEE Transactions on Information Forensics and Security*, 2 (3):438–449, 2007.

[134] W. Wang and H. Farid. Exposing digital forgeries in video by detecting double quantization. In *Proceedings of the 11th ACM workshop on Multimedia and security*, pages 39–48. ACM, 2009.

[135] W. Wang, X. Jiang, S. Wang, M. Wan, and T. Sun. Identifying video forgery process using optical flow. In *12th International Workshop on Digital-Forensics and Watermarking, IWDW 2013, Auckland, New Zealand, October 1-4, 2013*, pages 244–257. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-43886-2. doi:10.1007/978-3-662-43886-2_18.

[136] Y. K. Wang, C. T. Fan, K. Y. Cheng, and P. S. Deng. Real-time camera anomaly detection for real-world video surveillance. In *International Conference on Machine Learning and Cybernetics (ICMLC), 2011*, volume 4, pages 1520–1525. IEEE, July 2011. doi:10.1109/ICMLC.2011.6017032.

[137] J. Westerweel, D. Dabiri, and M. Gharib. The effect of a discrete window offset on the accuracy of cross-correlation analysis of digital PIV recordings. *Experiments in fluids*, 23(1):20–28, 1997.

[138] Y. Wu, X. Jiang, T. Sun, and W. Wang. Exposing video inter-frame forgery based on velocity field consistency. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2674–2678. IEEE, 2014.

[139] J. Xu, Y. Su, and X. You. Detection of video transcoding for digital forensics. In *2012 International Conference on Audio, Language and Image Processing (ICALIP)*, pages 160–164. IEEE, 2012.

[140] J. Xu, Y. Su, and Q. Liu. Detection of double MPEG-2 compression based on distributions of DCT coefficients. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(01):1354001, 2013.

[141] G. Yammine, E. Wige, and A. Kaup. Blind GOP structure analysis of MPEG-2 and H.264/AVC decoded video. In *Picture Coding Symposium (PCS), 2010*, pages 258–261. IEEE, 2010.

[142] J Yang, T. Huang, and L. Su. Using similarity analysis to detect frame duplication forgery in videos. *Multimedia Tools and Applications*, 75(4):1793–1811, Feb 2016. ISSN 1573-7721. doi:10.1007/s11042-014-2374-7.

[143] H. Yin, X. Jiao, X. Luo, and C. Yi. Sift-based camera tamper detection for video surveillance. In *25th Chinese Control and Decision Conference (CCDC), 2013*, pages 665–668. IEEE, May 2013. doi:10.1109/CCDC.2013.6561007.

[144] L. Yu, H. Wang, Q. Han, X. Niu, S. M. Yiu, J. Fang, and Z. Wang. Exposing frame deletion by detecting abrupt changes in video streams. *Neurocomputing*, 205(Supplement C):84 – 91, 2016. ISSN 0925-2312. doi:https://doi.org/10.1016/j.neucom.2016.03.051.

[145] YUV Sequence. http://trace.eas.asu.edu/yuv/. (Date last accessed 14-Dec-2016).

[146] J. Zhang, Y. Su, and M. Zhang. Exposing digital video forgery by ghost shadow artifact. In *Proceedings of the First ACM workshop on Multimedia in forensics*, pages 49–54. ACM, 2009.

[147] T. Zhang, Z. Yang, W. Jia, B. Yang, J. Yang, and X. He. A new method for violence detection in surveillance scenes. *Multimedia Tools and Applications*, 75(12):7327–7349, Jun 2016. ISSN 1573-7721. doi:10.1007/s11042-015-2648-8.

[148] Z. Zhang, J. Hou, Q. Ma, and Z. Li. Efficient video frame insertion and deletion detection based on inconsistency of correlations between local binary pattern coded frames. *Security and Communication Networks*, 8(2):311–320, 2015.

[149] D. N. Zhao, R. K. Wang, and Z. M. Lu. Inter-frame passive-blind forgery detection for video shot based on similarity analysis. *Multimedia Tools and Applications*, Mar 2018. ISSN 1573-7721. doi:10.1007/s11042-018-5791-1.

[150] X. Zhao, S. Wang, S. Li, and J. Li. Passive image-splicing detection by a 2-D noncausal markov model. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(2):185–199, Feb 2015. ISSN 1051-8215. doi:10.1109/TCSVT.2014.2347513.

[151] L. Zheng, T. Sun, and Y. Q. Shi. Inter-frame video forgery detection based on block-wise brightness variance descriptor. In *International Workshop on Digital Watermarking*, pages 18–30. Springer International Publishing, 2014. ISBN 978-3-319-19321-2. doi:10.1007/978-3-319-19321-2_2.