UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

PH.D. THESIS
IN
INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

A NEW TECHNIQUE FOR VIDEO COPY-MOVE FORGERY DETECTION

LUCA D'AMIANO

TUTOR: Prof. Giovanni POGGI

COORDINATOR: Prof. Daniele RICCIO

XXX CICLO

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

“What is the most resilient parasite?
Bacteria? A virus? An intestinal worm?
An idea.
Resilient... highly contagious.
Once an idea has taken hold of the brain
it’s almost impossible to eradicate.
An idea that is fully formed
- fully understood - that sticks;
right in there somewhere.”

Contents

List of Figures **iii**

Introduction **1**

1 Detection of Image and Video Copy-Move Forgeries **3**
1.1 A Short History of Image and Video Forgery 3
1.2 A Classification of Image and Video Forgeries 12
1.3 Overview on Image Copy-Move Forgery Detection 18
1.3.1 A Dense-Field Technique for Image Copy-Move Forgery Detection 22
1.4 Overview on Video Copy-Move Forgery Detection 24
1.4.1 A Method for Detection of Block Duplications in Videos 26

2 Feature Extraction **31**
2.1 Zernike Moments 33
2.2 3D Flip-Invariant Features 36

3 Matching Algorithm **39**
3.1 PatchMatch 41
3.2 Modified PatchMatch 42
3.2.1 Adapting PatchMatch to video 43
3.3 PatchMatch with Multiresolution 44

4 Post Processing **49**
4.1 Dense Linear Fitting 49
4.1.1 Morphological Operations 51
4.2 Removing False Alarms 53

5 Experimental Results **57**
5.1 The GRIP Dataset 57
5.2 Performance Assessment 62
5.2.1 Numerical results 66
5.3 Complexity 71
5.4 A real-world case: the Varoufakis video 77

Conclusion **81**

Bibliography **83**

List of Figures

1.1 Tampered picture of President Abraham Lincoln 4
1.2 Tampered picture of General Ulysses Grant 5
1.3 Tampered picture of Dictator Benito Mussolini 5
1.4 Tampered pictures in the show business 6
1.5 Kerry-Fonda tampered picture 7
1.6 Tampered picture of English soldier in Iraq 7
1.7 Tampered picture of city bombing in Lebanon 8
1.8 Tampered picture of iranian missiles 8
1.9 Jedi Kittens tampered video 9
1.10 Varoufakis' foregery 10
1.11 Face-to-Face tampered video 11
1.12 Example of image copy-move. 13
1.13 Image inpainting. 14
1.14 Intra-frame Video copy-move 15
1.15 Additive and occlusive video copy-moves 15
1.16 Copy-moves with geometric transformations 17
1.17 Examples from FAU and GRIP databases 20
1.18 Experimental results for image CMFD techniques 21
1.19 Modified PatchMatch: first-order predictors 23
1.20 Video forgery detection: classification scheme 24
1.21 Method proposed in [7]: trends of $c^t_{B^n_m}$ 28
1.22 Method proposed in [7]: examples of detections 29

2.1 Examples of radial profiles 34
2.2 Examples of sampling grids 35
2.3 Feature 3D explanation 37

3.1 PM vs KDtree: computational time 40
3.2 PatchMatch 3D: propagation in time dimesion 43

3.3 Block diagram of the proposed video copy-move detector . . . 45
3.4 Video partitions for PM parallel computing 47

4.1 Post-processing steps 52
4.2 Removing random false matches 53
4.3 Detection Maps: DUAL vs NO-DUAL 55

5.1 GRIP dataset: videos from #1 to #5 58
5.2 GRIP dataset: videos from #6 to #10 59
5.3 GRIP dataset: videos from #11 to #15 60
5.4 Examples of ground truth and detection map 62
5.5 Examples of 3D detection maps 68
5.6 Trends of detection variables' distributions 69
5.7 ROCs comparison between proposed technique and [7] 71
5.8 ROCs: DUAL vs NO-DUAL 72
5.9 Detection maps: examples for plain CMs 73
5.10 Detection maps: examples for plain pristine 74
5.11 Detection maps: examples for CMs with rotation 75
5.12 Detection maps: examples with compression 76
5.13 Computational cost of the proposed algorithm 77
5.14 Varoufakis' videos 78
5.15 Detection in Varoufakis videos 79

Introduction

Nowadays, anyone can easily modify the appearance and content of digital images by means of powerful and easy-to-use editing tools such as Adobe Photoshop, Paintshop Pro or GIMP. This becomes more and more true also for digital videos. Powerful and widespread tools exist for video editing, like Adobe After Effects and Premiere Pro, which allow users to perform a number of video manipulations. Most of the times, these have the only purpose of improving the quality of videos or their appeal. Sometimes, however, they are not so innocent, aiming at falsifying evidence in court, perpetrating frauds or discrediting people. Therefore, in the scientific community there is an increasing activity towards the design of efficient and reliable techniques for the detection and localization of video forgeries [73].

One of the simplest, yet effective, video manipulations consists in copy-moving a video object from a source location to one or more target locations. This attack can be additive, when a semantically relevant object is copy-moved, or occlusive, when part of a *background* region is copy-moved to hide a foreground object. In both cases, a number of tricks can be used to reduce the attack detectability, like rotating or resizing the object, flipping it along the temporal axis, adding noise, etc. However, additive attacks provide a number of clues to the investigator, from the very same presence of visually similar objects which can raise the attention of the observer, to the presence of salient details (keypoints) which can be exploited to match the clones. Occlusive copy-moves, instead, offer no such clues, and are in fact much more difficult to detect.

In this thesis a new algorithm for the reliable detection and localization of video copy-move forgeries is proposed, taking inspiration from the technique proposed in [26] for still images. To reliably detect both additive and occlusive copy-moves, we use a *dense-field* approach, based on the matching of suitable features, computed on a spatio-temporal grid and invariant to various spatial,

temporal, and intensity transformations, which guarantee robustness to several post-processing operations. To limit complexity in the matching phase, a suitable video-oriented version of PatchMatch [3, 4] is used, with a multiresolution search strategy. PatchMatch allows us to build a nearest-neighbor field (NNF), connecting each feature with its best-matching [26, 32, 33]. Finally, the NNF is post-processed to single out areas with coherent spatio-temporal displacement as candidate copy-moves.

To the best of our knowledge, this is the first algorithm for video copy-move detection based on a dense-field approach. Thanks to this choice, it is able to detect and localize reliably both additive and occlusive copy-moves. Moreover, despite the huge computational challenge intrinsic in copy-move video detection, it runs in a reasonable time, allowing effective forensics analyses. As said before, it extends the detector proposed in [26] for still images, introducing, however, a number of innovations to deal efficiently with videos. The main technical innovations of this work consist in *i)* the definition of a new flip-invariant 3D feature based on the Zernike moments; *ii)* the design of a fast version of the matching algorithm, based on multi-scale processing and parallel implementation; *iii)* the introduction of a new criterion in the post-processing phase to tell apart copy-moves from false matches. The algorithm is available online (www.grip.unina.it) to guarantee research reproducibility and to enable other researchers to innovate upon this basis. A further contribution of this work concerns performance assessment, which relies on a new dataset, designed ad hoc, and including realistic copy-moves, both additive and occlusive, in a wide variety of challenging situations. The experimental results show that the proposed method is able to detect and localize with good accuracy video copy-moves, even in adverse conditions.

In the rest of the thesis, **Chapter 1** provides an overview on image and video forgery detection, with special focus on copy-move attacks, and analyzes the state of the art. The following chapters describe the proposed algorithm, analyzing in turn feature extraction, **Chapter 2**, matching, **Chapter 3**, and post-processing **Chapter 4**. **Chapter 5** presents the experimental setting and analyzes numerical results, leading to the final conclusions.

Chapter 1

Detection of Image and Video Copy-Move Forgeries

This chapter provides the introductory material necessary for the full understanding of the proposed method described later on. After an historical tour on image and video forgery, these attacks are classified systematically, before going in more depth on the problem of detecting and localizing copy-move attacks, both in image and videos.

1.1 A Short History of Image and Video Forgery

Photography is an extraordinary communication tool. Its ability to convey information has been accompanying us in many aspects of our lives for two centuries. It is often said that *an image is worth a thousand words*, and also that *seeing is believing*. However, to keep relying on proverbs, *all that glitters is not gold*, and more and more often pictures are used to convey false or distorted information.

The first attempts to obtain some forms of photography date back to the early 19th century, although the photographic process reached maturity only around the middle of the century when taking a picture became possible without a long exposure of the subject. As time went on, photography began to play a fundamental role not only for entertainment, but also in a number of practical applications, especially in the commercial, journalistic and political fields. Moreover, due to their ability to provide rich information, and their perceived reliability, photos started very soon to be used as evidence in courtrooms.

The continuous technological progress led to many innovations throughout

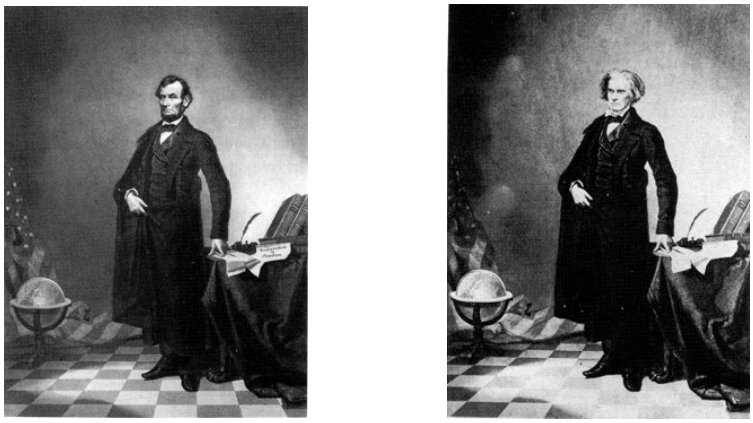


Figure 1.1: Tampered picture of President Abraham Lincoln: on the left the forged image, on the right the pristine image.

these two century, however it seems safe to say that the advent of digital photography brought about a huge shift of paradigm. Taking and sharing photos has become straightforward, and we are now surrounded and overwhelmed by billions of pictures, also thanks to the rapid diffusion of social networks. Indeed, a fundamental feature of these new platforms is the opportunity of easily sharing multimedia material, especially photographs.

However, with images acquired and stored in digital form, the fast advances of image processing methods and tools has made very easy to modify their content. Actually, image manipulation has quite a long history, but this *art* was once limited to a small number of very skilled individuals. With modern image editing tools, instead, anyone can easily manipulate images, threatening the very same use of these pieces of information for sensitive applications, especially as investigative tools and evidence in court. In this section we are going to expose, through a historical excursus, some common kinds of falsifications that can be carried out on digital images.

It is not advisable to trust every photograph we see. Indeed, pictures were modified already a few decades after the birth of the first photograph by Niepce in 1814. One of the first known manipulated images dates back to 1860 and portrays the United States President Abraham Lincoln (Fig.1.1). Lincoln's head was put on the body of another man, the politician John Calhoun, in order to obtain a more dramatic, and almost heroic, style. Another famous fake is a photograph of General Ulysses S. Grant, showing him in front of his troops on the City Point (Virginia) battlefield in 1864, during the American Civil War.

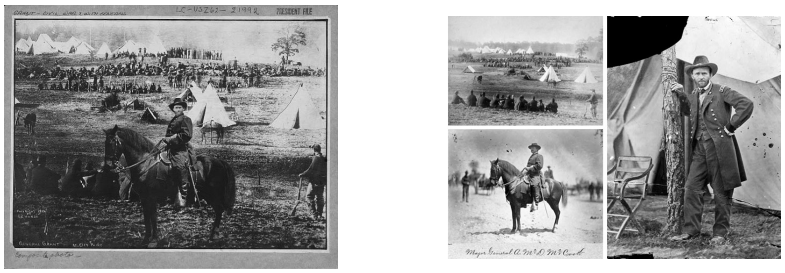


Figure 1.2: Tampered picture of General Ulysses Grant: on the right the pristine images used to make the forged image on the left.

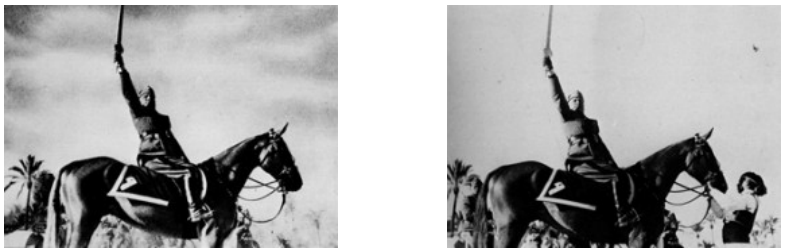


Figure 1.3: Tampered picture of Dictator Benito Mussolini: on the left forged image, on the right pristine image.

An accurate research of the Congress Library has revealed that this image is an excellent composition of three different photos, as shown in Fig.1.2.

Jumping to the twentieth century, the 1942 picture of Fig.1.3, featuring the Italian dictator Benito Mussolini, was modified by removing the person who held the horse to create a heroic atmosphere.

With digital cameras and software editing tools, it has become very easy to modify images. It has been estimated that almost the 10% of the colored pictures published in the United States of America in 1989 were already falsified and defaced. In 1989, the cover of Tv Guide showed the famous anchorwoman Oprah Winfrey, but actually it was a fake: Oprah's head was put on the body of another actress. The false picture was created without the actress' authorization and she discovered the trick recognizing her dress (Fig.1.4). In 2005, Newsweek's cover illustrated a photograph of Martha Stewart with a caption calling to mind her weight loss due to the prison term. The picture was not real and Stewart's head was actually placed on a model's slimmer body (Fig.1.4).

During the 2004 U.S. presidential campaign, a photo was diffused portraying Senator John Kerry was, candidate for the Democratic party, next to the

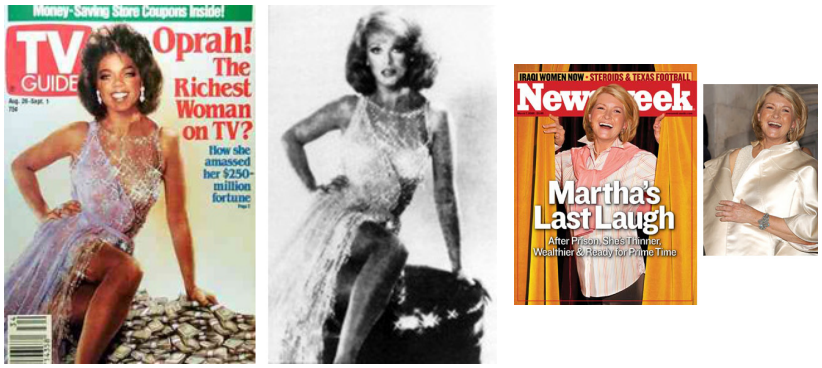


Figure 1.4: Tampered pictures of Oprah Winfrey (left) and Martha Stewart's (right).

activist Jane Fonda during an assembly against the war in Vietnam (Fig.1.5). But, again, it was a collage of two different photos, later retrieved, when the damage to Kerry campaign was done and irrecoverable.

There are many manipulations also in more sensitive contexts than politics or show businesses, it is the case of war's pictures. In April 2003, in fact, a photo was published showing an English soldier in Basra, Iraq, who commanded Iraqi civilians to bend down. This picture, taken by Brian Walski, was made public on the Los Angeles Times' cover after the invasion of Iraq by United States of America. When the publisher found out that the image was a juxtaposition of two different photos, he did not hesitate to dismiss the photographer although he had thirty years of experience in his line of work (Fig.1.6). Another example dates back to August 2006 when Reuters Agency published on its website a picture, taken by the Lebanese photographer Adnan Haji, that portrayed a Lebanese city bombarded by Israeli. The image was, successively, deleted when a manipulation was discovered, which increased the smoke due to the bombing in order to overstate the impact over the city (Fig.1.7). Reuters removed more than 1000 pictures taken by Haji from his archive.

In 2008 an Iranian missile was digitally added to an image in order to conceal a missile on the ground that did not fire. The forged image appeared on the front page of many major newspapers (Fig.1.8).

All the cases previously exposed are a little part of an endless sequence of photos that have been manipulated and spread through history[1]. In each case

[1]The link **http://www.fourandsix.com/photo-tampering-history/** provides many other examples of forged images

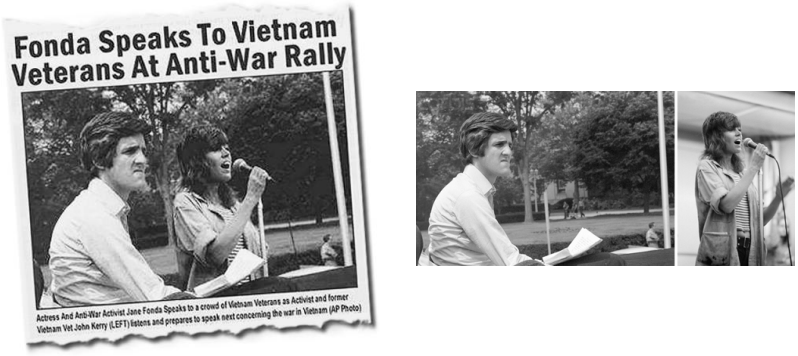


Figure 1.5: Tampered picture obtained by the juxtaposition of John Kerry and Jane Fonda images: on the right the pristine images used to fabricate the forged image on the left.

Figure 1.6: Tampered picture of an English soldier in Iraq: on the right pristine images used to make forged image on the left.

Figure 1.7: Tampered picture of city bombing in Lebanon: on the right forged image, on the left pristine image.

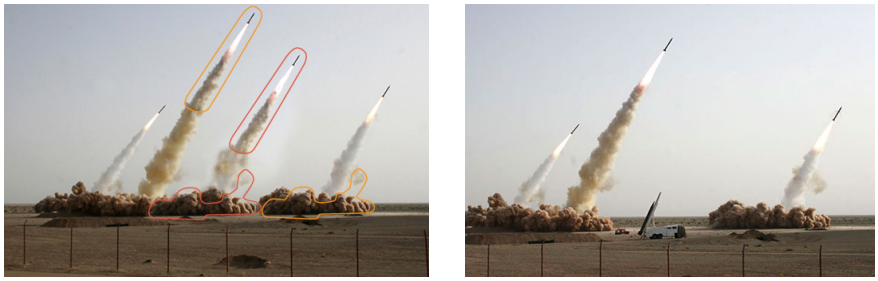


Figure 1.8: Tampered picture of iranian missiles. Right: the forged image with an added missile. left: the pristine image.

the aim is to spread false news or to smear a particular event, a person or a nation. Studies have shown that modified images stick in the mind for a long time and they are also able to create or alter memories of who is looking. Few days before the Presidential elections in 2004, a voter was asked for whom he had voted and why. Among the many reasons to justify his vote for George W. Bush, the man said he could not forget the picture of John Kerry and Jane Fonda together even though he knew it was a fake. During a research, original and false images of memorable events were shown to the participants. The modified photos, portraying a larger crowd or more violence, succeeded to change participants' memories tied to that event, a proof that photography is one of the new media forms able to change men's perception.

There are many different software tools able to realize these types of manipulations. Starting with programs created for image visualization, that often allow to correct some imperfections, modify colors or resize the number of pixels, until the latest tools specifically projected to make more advanced forgeries as Photoshop, Gimp, Paint and so on.

Figure 1.9: Some frames from the Jedi Kittens tampered video

Unlike for digital images, it is not so easy to find documented cases of tampered videos. This is because it is much more difficult to manipulate a video data structure changing its informative content and leaving it credible. Even if video editing tools are becoming more and more user-friendly, manipulating the video content in order to modify its information is still a job for professionals. Probably, this is one of the reasons why video was considered as an infallible instrument to show the evidence of a fact until about a decade ago [95].

In the last few years, however, there has been a sharp increase in the number and quality of tools used for video manipulation (Adobe Premiere, Adobe AfterEffects, Photoshop, Cinelerra, Lightworks etc.), many scientific papers have been written to propose powerful techniques for video manipulation [60, 83, 30, 92, 57, 82, 48], and the phenomenon is growing fast. In some cases, it is possible to recognize the video manipulation by visual inspection, since its contents appears to be unrealistic; in other cases the video content looks so natural that only forensic techniques may be able to expose the forgery. Just like for digital images, video manipulations can be realized for various aims, both benign and malevolent.

Notable examples of the first type are the videos realized by Zach King, a video artist very popular on social media. In 2011 King posted on YouTube a video called *Jedi Kittens*[2] featuring two cats fighting with laser swords (see Fig.1.9). The video had been obviously tampered with, but the manipulation was performed with great skill, obtaining a great impact. Within a few days from its publication it reached millions of visualizations.

Turning to malevolent tampering, it is worth mentioning the Varoufakis case, which made the headlines in 2015. Fig.1.10 shows frames taken from

[2]https://www.youtube.com/watch?v=NtgtMQwr3Ko

Figure 1.10: Frames taken from the three Varoufakis videos. From top to down: #1, sticking middle finger, #2, arm down, #3, victory sign.

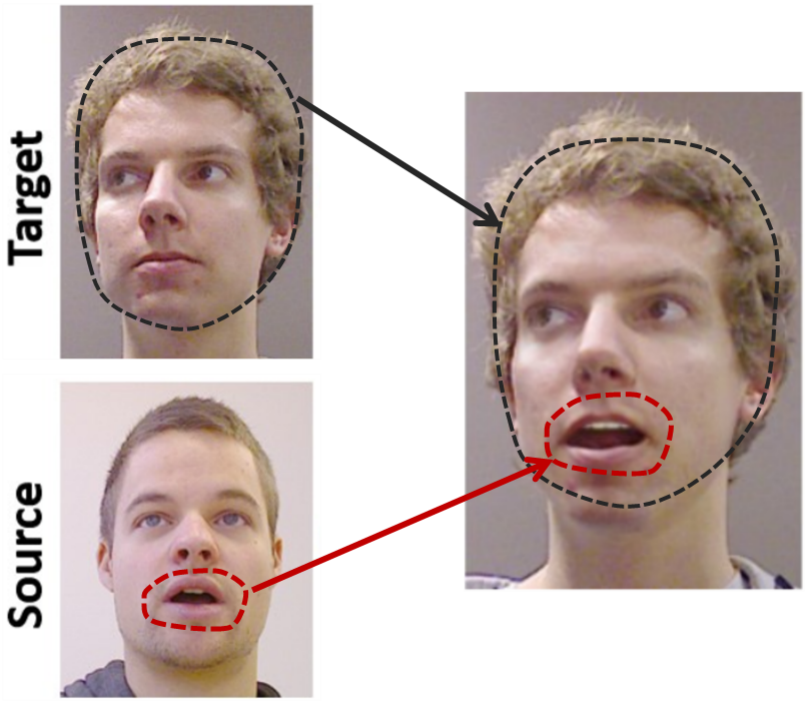


Figure 1.11: A face-to-Face tampered video featuring the manipulation of lips movement. Frames taken from source and target videos (left) are used to create the manipulated video on the right.

three different versions of the same video, all allegedly original, portraying the Greek economist and politician Yanis Varoufakis during a public speech. In the first version, Varoufakis raises his arm and shows the middle finger while mentioning Germany; in the second one, he does not even raise his arm; in the third one, he raises his arm and shows middle and index fingers in the victory sign. The diffusion of the first video was followed by a heated media debate: Varoufakis immediately disproved the allegations. Obviously, two of the videos are well-crafted fakes, and it is not easy to find out which one by visual inspection, and not even by forensic analyses. In the paragraph 5.4 there is a detailed analysis of the three videos, conducted with the algorithm proposed in this thesis, with experimental proofs that partially dissipate the doubts.

The former example, however, was still rather naive. A much more

threatening attack is represented by software tools such as Face-to-Face. In 2015, researchers of the Max-Planck-Institute for Informatics of the Erlangen-Nuremberg University (D) and of the Stanford University (USA), introduced a new technique that allows to manipulate very effectively a person's face in a video in real time [104] [105]. The authors demonstrate a possible application of this technique. They take a source video, filmed in laboratory, and a target video, typically downloaded from social media and show that it is possible to replace the face's labial in the target video with the lips' movement of the source video (Fig.1.11). In other words, the target video can be manipulated so that the person appears to pronounce some specific words, which of course can be easily modified as well. It is not hard to imagine the damage that such a tool could bring to a person's image if used malevolently.

1.2 A Classification of Image and Video Forgeries

There are a number of ways to manipulated an image content, involving both local and global processes, and possibly combined with one another. We briefly describe, here, the most important attacks, that is splicing, copy-move, and inpainting.

Splicing: consists in pasting on the target image an object taken from another (source) image. The source can be both a natural or computer-generated image.

Copy-Move: differs from splicing because source and target images coincide. That is, the object is copied from a position of the image and moved somewhere else.

Inpainting: was proposed originally to restore parts of images damaged by scratches or other impairments, by filling the voids in correspondence of the damaged part. However, it has soon be employed to remove unwanted objects from the image, extending the background to hide the traces.

The forgeries shown in figures 1.1 through 1.6 are all splicing, while those of figures 1.7 and 1.8 are copy-moves. Both attacks can be additive and occlusive. Assuming that the image can be divided in foreground objects and a background, an additive forgery is obtained by *inserting* a foreground object in the image, taken either form another (splicing) or the same (copy-move) image. Instead, an occlusive forgery aims at *hiding* a foreground object by means of some background areas, which is almost always taken from the same image to obtain a more realistic effect. Under this point of view, inpainting is

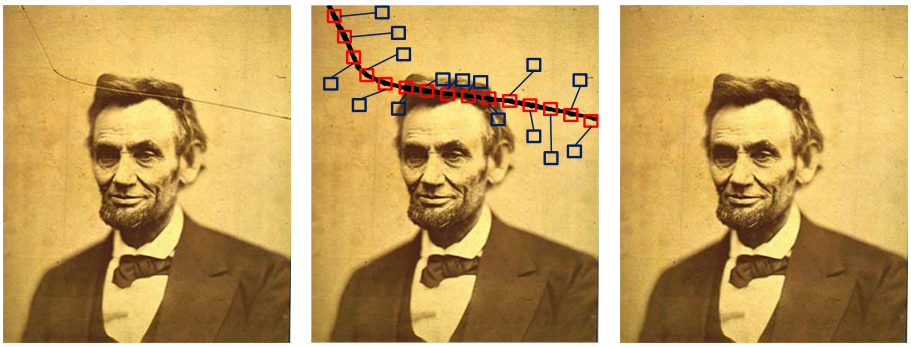
Figure 1.12: Example of image copy-move. Pristine image (top left), forged image (top right). A branch of the tree from the top right side of the image is copied and pasted in the middle side of the image (as highlighted by the image in the bottom)

always occlusive. Unlike copy-move, it does not cover the foreground image by means of a single region, but rather juxtaposing a large number of small blocks (exemplar-based inpainting) or else diffusing the background to fill the unwanted region (diffusive inpainting).

Fig.1.12 shows a clear example of additive copy-move, while Fig.1.13 illustrates the phases of a diffusive inpainting.

Turning to videos, the same type of attacks exist, with obvious differences due to the three-dimensional (or 2D+t) nature of the objects. However, a further distinction is due between *Inter-frame* and *Intra-frame* forgeries.

We talk of **Inter-frame forgeries** when the manipulation affects groups of video frames taken as a whole. Examples of Inter-frame forgery are the removal or insertion of whole video frames, or the cloning of a set of video frames from one temporal location to another (inter-frame copy-move).



(a) (b) (c)

Figure 1.13: Example of image inpainting. Original photo with a scratch (a), inpainting mask (b), inpainted image (c). The scratch region will be filled with content taken from the surrounding area.

In **Intra-frame forgeries**, instead, the manipulation affects only a part of a frame or group of frames (see Fig.1.14). This technique is therefore used to remove or add video objects. Partial copy-move (only a part of the frame is manipulated) and inpainting techniques can be included in this category.

Inter-frame techniques are simpler to perform but less flexible allowing a restricted number of manipulations. Intra-frame techniques, instead, allow more sophisticated manipulations. For example, it is possible to remove a person from a surveillance video replacing it with suitable material taken from the same or other videos [113]. These techniques are more difficult to perform, but allow for more flexible and subtle content modifications. Moreover, if properly carried out [47, 46], they can be quite challenging to detect, as they leave no obvious traces in the video temporal structure. As already mentioned, these attacks can be additive, when a video object of interest is inserted anew in the target video, or occlusive, when an object is deleted from the video, through inpainting or by copying background over it. Fig.1.15 shows examples of both situations.

A more systematic review of intra-frame attacks is summarized in Tab.1.1. Additive forgeries may be performed by pasting a video object taken either from another video (splicing) or from the same video (copy-move). The first case is certainly more interesting, however it can be hardly performed without leaving a long trail of clues. In fact, objects taken from a different source are very likely to have statistical, physical and semantic properties incompatible

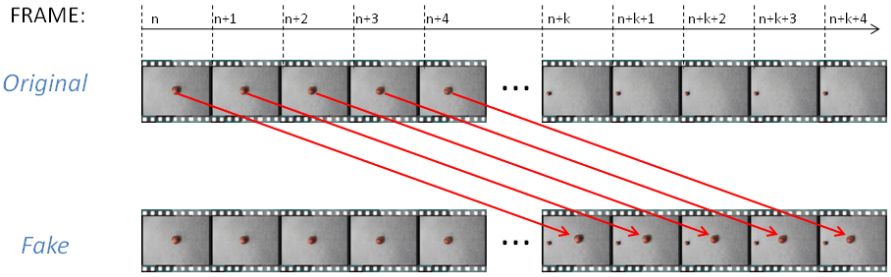
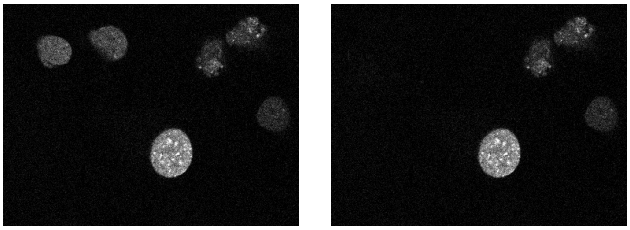


Figure 1.14: Additive intra-frame video copy-move. A ball from the source video is copied and pasted in another spatio-temporal position of the same video.

(a) Original frame (b) Additive forgery



(a) Original frame (b) Occlusive forgery

Figure 1.15: Additive (top) and occlusive (bottom) video copy-moves. Cell counting videos, like those shown in the bottom, can be easily manipulated to commit scientific frauds. Detection may be quite difficult, especially in the occlusive case, due to the lack of salient keypoints.

attack	action	target area	source video	insertion artifacts	physical inconsistencies	statistical inconsistencies
additive splicing	copy	object	different	✓	✓	✓
additive copy-move	〃	〃	same	✓	–	–
occlusive splicing	〃	background	different	–	–	✓
occlusive copy-move	〃	〃	same	–	–	–
in-painting	synthesize	〃	same	–	–	✓

Table 1.1: Video forgery attacks' properties. When new objects are inserted in a video (lines 1 and 2) artifacts at the boundaries easily appear. If objects are taken from another video (line 1) physical and statistical inconsistencies are also likely, while copy-moved objects in the same video (line 2) may originate suspicious multiple clones. Hiding objects with background (lines 3-5) is easier, but finding the right textured cover in other videos (3) may be hard. Moreover, splicing (line 3) and in-painting (line 5) always produce statistical inconsistencies and in-painting works only for small areas. With some care, occlusive copy-moves (line 4) can be performed without leaving obvious traces.

with those of the target video. Copy-move, whenever applicable, is definitely preferable, as it can be implemented leaving little visible traces. On the down side, repeated instances of the same foreground object may easily raise the attention and suspects of viewers. Occlusive forgeries are generally simpler to perform, since background areas abound, and do not raise much attention. Using material taken from other videos for this purpose (splicing) makes little sense, and may be difficult when the background is textured. The simplest way to hide a subject is by copy-moving on it parts of background taken from other frames of the same video. A possible alternative is background synthesis (inpainting) but it applies only to objects relatively small in space or time, otherwise the quality of the synthesis can degrade significantly. In summary, copy-move is a highly effective editing operation that is at the same time technically also quite straightforward to perform. As such, it is probably the manipulation attack with the best cost-effect ratio. A carefully executed copy-move can easily fool visual scrutiny, especially in the occlusive case. Moreover, it may elude forensic tools looking for statistical inconsistencies,

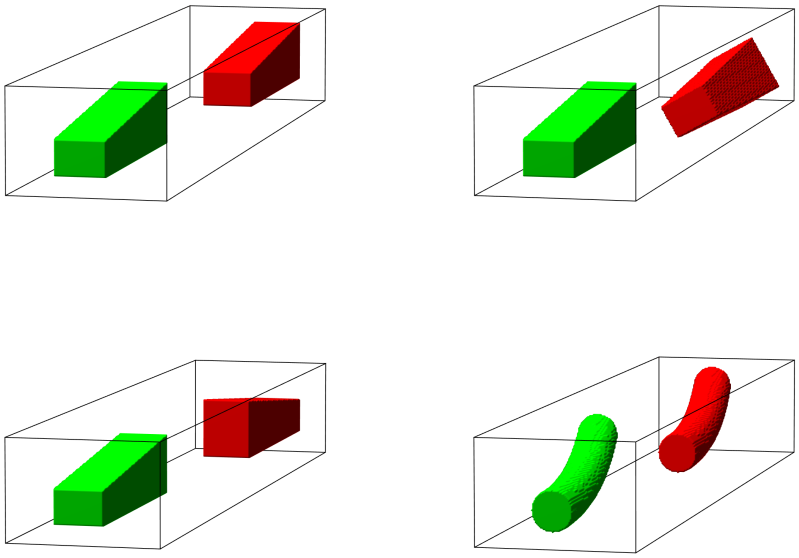


Figure 1.16: Copy-moves with geometric transformations. In green and red are shown the spatio-temporal supports of source and copied video objects, respectively, for several types of copy-moves: plain (top left), with rotation (top right) , with temporal flip (bottom left), plain with moving object (bottom right).

since the copied object comes from the video itself.

From this analysis, it is clear that video copy-moves represent by far the easiest and most effective way to perform a video forgery. In particular, well-crafted occlusive copy-moves cannot be spotted by visual inspection, and the analyst must rely exclusively on automatic tools. However, if a video object is copied *as is* from one location to another, it is very simple to detect the duplication, even by trivial lexicographic ordering of video patches. For this reason, copy-moves are typically accompanied by some further processing which guarantees that source and target objects are not identical, and hence much more difficult to detect (Fig.1.16). Among the most common such manipulations we can list rotation, resizing, change of brightness, noise addition, temporal flipping. The first three are routinely used also to adapt the copied

video object to the new context, which may differ from the original one under many respects. In any case, they have the effect of changing the copy with respect to the source, making naive approaches worthless. The addition of noise is enacted on purpose for the same reason and may regard only the clone or the whole video. In the first case, the clone will exhibit an anomalous level of noise, a clue for the analyst, in the second one the video quality will decrease. Temporal flipping is also made on purpose: by reversing the order of the frames in the clone the object moves backwards and can be hardly associated with he source object. For natural objects, the movement may appear unnatural and be spotted easily, but often this is not the case as for the cell counting video of Fig.1.15 bottom. To all these action, usually further processing steps are added, like the smoothing of boundaries, to avoid sharp transitions from the object boundaries to the background, and some forms of compression which, distorting the whole video, tends to further hide the traces of manipulation.

1.3 Overview on Image Copy-Move Forgery Detection

To face the growing threat of image forgery, a large number of methods have been proposed in recent years in the scientific literature. Following [38], they can be grouped in five large families following distinct approaches. *Pixel-based* techniques are based on statistical analyses of the data, both in the original domain (pixels) and in some transform domain [108, 24, 27]. For example [102] detects possible forgeries based on blur inconsistencies in the image, [54] relies on the traces left by resampling, [20] detects cloned regions based on the matching of local invariant features, while [28, 29] train neural networks to find anomalies in the digital image. *Format-based* methods detect forgeries based on the traces left by lossy compression schemes. For example, the methods proposed in [69, 13, 8] detect traces of double JPEG compression or other JPEG-related artifacts. *Camera-based* techniques [118, 42, 85, 40, 71, 11, 16, 25, 17, 18, 15] rely on the traces left in the image during the acquisition phase, which can be regarded as image signatures. *Physics-based* and *geometric-based* techniques [52, 53] look for physical or geometrical inconsistencies of the objects present in the image, such as inconsistent lighting, shadows or geometric features in the scene.

We do not try to explore this large body of literature, here, referring the reader to a few recent reviews [84, 10, 87, 97] for more information. Instead, in the following we analyze in more depth image copy-move forgery detection methods, describing in more detail the work of Cozzolino *et al.* to which this

thesis work is inspired. Then, in the following section, we will do the same for the video case.

In the last few years, a large number of techniques have been proposed for the detection and localization of copy-move forgeries in digital images [20]. Virtually all such techniques comprise three major steps: *i)* feature extraction, *ii)* matching, and *iii)* post-processing. In the first step a suitable feature is associated with each pixel of interest. Based on such features, each pixel is then linked with its best match over the image, generating a field of offsets. Finally, this field is processed to single out regularities which points at possible copy-moves.

Some techniques, e.g. [80, 1, 121], operate only on a small set of salient keypoints, characterized through well-known local descriptors, such as SIFT or LBP. This approach is computationally efficient, but it fails completely if no keypoint is associated with the forgery, as in the common case of occlusive copy-moves over a smooth background [20, 26].

Techniques based on dense sampling are much more reliable. Their main issue is complexity, since all pixels are involved in the three phases of feature extraction, matching, and post-processing. To reduce computation, compact features are extracted, typically through some transforms, like DCT [41], wavelet [75], PCA [72] or SVD [120]. By so doing, a good robustness is also obtained with respect to intensity distortions, originated for example by JPEG compression or blurring. Instead, to deal with geometric distortions due to rotated or rescaled copy-moves, specific invariant features are needed. The Zernike moments and the polar sine and cosine transforms have been used [89, 64, 63] to obtain rotation invariance, while for scale-invariance the Fourier-Mellin Transform with log-polar sampling has been considered [6, 114] (Chapter 2).

Feature extraction, however, is only part of the problem. Exhaustive search of the best matching (nearest neighbor) feature is prohibitively complex, and faster techniques must be devised to produce the offset field in a reasonable time. To this end, approximate search strategies have been used, such as kd-tree search, in [62, 20], or locality sensitive hashing, in [89, 64]. Nonetheless, computing the nearest-neighbor field keeps being too slow for the large images generated by today's cameras. A much better result can be obtained, however, by exploiting the strong regularity exhibited by the NNFs of natural images, where similar offsets are often associated with neighboring pixels. This is done in [22] and [26], where the offset field is computed by means of a suitably modified version of PatchMatch [3, 4], a fast randomized search technique

Figure 1.17: Forged images (on the left) from the FAU database (top) and the GRIP database (bottom) with relatives ground truth binary maps (on the right).

specifically tailored to the properties of images (Chapter 3).

To assess the performance of image copy-move detection techniques, several dedicated datasets have been designed. One of the most popular is the FAU dataset[3], proposed by Christlein *et al.* in their review paper [20], which comprises 48 realistic copy-moves divided in three classes: smooth, rough and structured. Also useful is the GRIP dataset[4], proposed by Cozzolino *et al.* [26], comprising 80 accurate realistic copy-moves. Fig. 1.17 shows examples from both datasets. In Fig.1.18, instead, we show some example results computed experimentally on the GRIP dataset. The performance is in terms of F-measure, and various cases of interest are considered besides plain copy-moves, including noise addition, rotation, resizing, and JPEG compression.

[3] http://www5.cs.fau.de/
[4] http://www.grip.unina.it

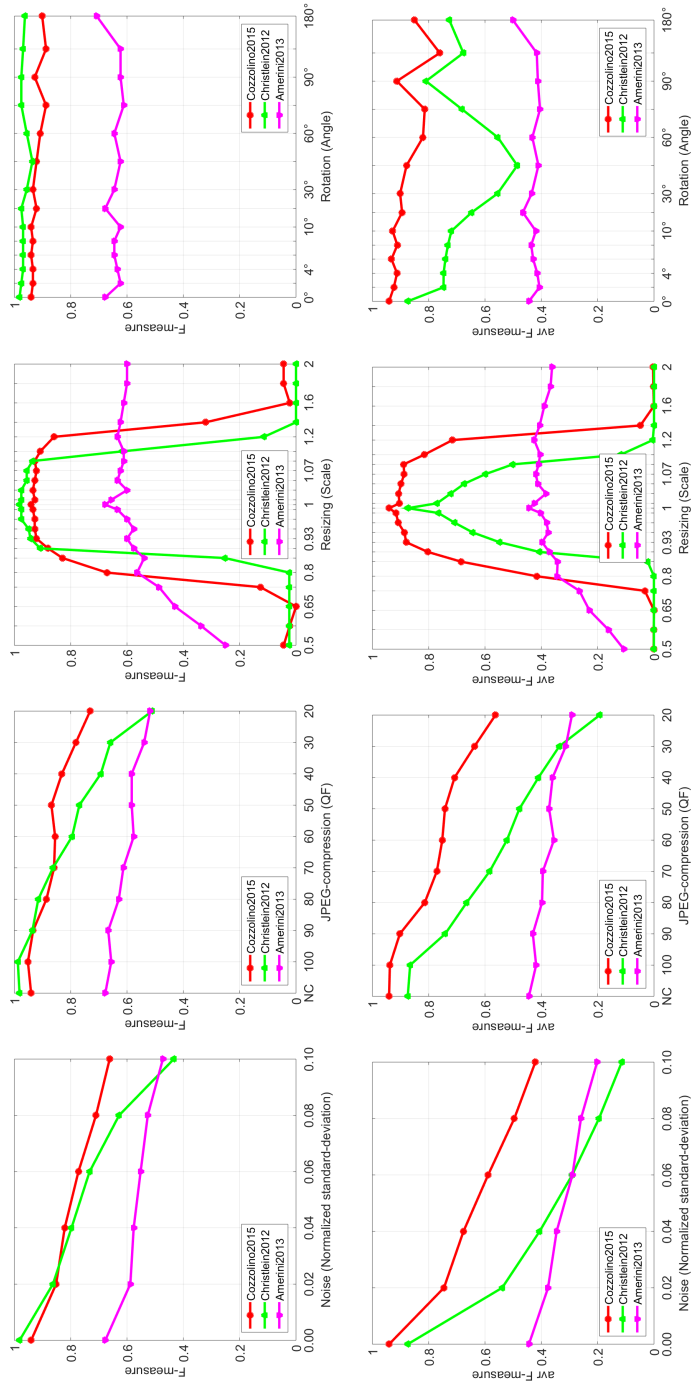


Figure 1.18: Top: Image-level F-measure curves for *Christlein2012*, *Amerini2013* and *Cozzolino2015*'s techniques. Bottom: Pixel-level F-measure curves for *Christlein2012*, *Amerini2013* and *Cozzolino2015*'s techniques.

1.3.1 A Dense-Field Technique for Image Copy-Move Forgery Detection

In this paragraph we briefly describe the method for image copy-move detection and localization, proposed in [26], on which this thesis work relies. Thanks to the use of rotation-invariant and robust features, copy-moves are reliably detected even in the presence of various forms of intensity and geometric distortion. Efficiency is ensured by using a suitably modified version of PatchMatch for the offset field computation and a fast *ad hoc* post-processing to remove false matches.

Let $I(\rho, \theta)$ be the input image in polar coordinates, with $\rho \in [0, \infty]$ and $\theta \in [0, 2\pi]$, and let

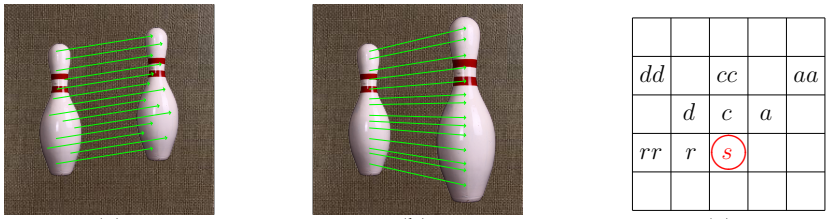
$$K_{n,m}(\rho, \theta) = R_{n,m}(\rho) \frac{1}{\sqrt{2\pi}} e^{jm\theta} \tag{1.1}$$

be a kernel function obtained as the product of a radial profile $R_{n,m}(\rho)$ and a circular harmonic. By projecting the image over the kernel we obtain the feature

$$f(n, m) = \int_0^\infty \rho R^*_{n,m}(\rho) \times \left[\frac{1}{\sqrt{2\pi}} \int_0^{2\pi} I(\rho, \theta) e^{-jm\theta} \mathrm{d}\theta \right] \mathrm{d}\rho \tag{1.2}$$

By choosing the Zernike orthonormal radial functions [103] $f(n, m)$ turns out to be the Zernike moment of order (n, m) of the image. Note that the integral in square brackets is the Fourier series of $I(\rho, \theta)$ along the angle coordinate, and its magnitude is invariant to rotations of the image I. Therefore, by selecting as features the magnitude of Zernike moments we guarantee rotation invariance. In addition, if only a few low-order moments are used, a compact feature vector is obtained, robust to intensity distortions, which are mostly of high-pass nature.

To compute the offset field efficiently authors resort to PatchMatch. However, the basic version of the algorithm is designed for patchwise constant offset fields, a model appropriate for rigid copy-moves, as in Fig.1.19(a), while rotated and resized copy-moves give rise to linearly varying offsets, as in Fig.1.19(b). A generalized version of PatchMatch was proposed in [4] to deal with this problem. Unfortunately, it works only on image patches (not compact features) and is significantly more complex than the basic version. A much simpler modification was proposed in [22], adding first-order predictors to the zero-order predictors used in PatchMatch, so as to deal effectively also



(a) (b) (c)

Figure 1.19: Modified PatchMatch. With rigid copy-moves (left) clones are connected by a constant offset field. In the presence of rotation and/or resizing (center) clones are connected by a linearly varying offset field. PatchMatch (right) uses zero-order predictors of the offset, based on neighboring pixels (r, c, d, a) on the same row, column, or diagonal as the target (s). The modified version uses also first-order predictors, involving neighbors farther apart (rr, cc, dd, aa) so as to follow linear variations.

with linear offset fields. With reference to Fig.1.19(c), a zero order prediction of the offset $\delta(s)$ at site s is given by

$$\widetilde{\delta}^{0x}(s) = \delta(x), \qquad x \in \{r, d, c, a\} \tag{1.3}$$

that is, the offset is predicted as being equal to the offset of the neighbor on the same row, column, diagonal or antidiagonal. Adding first-order predictors

$$\widetilde{\delta}^{1x}(s) = 2\delta(x) - \delta(xx) \tag{1.4}$$

we take into account linear variations of the offset along the same four directions. Eventually, we obtain the enlarged set of predicted offsets

$$\begin{aligned} \Delta^P(s) \quad = \quad & \{\delta(s), \widetilde{\delta}^{0r}(s), \widetilde{\delta}^{0d}(s), \widetilde{\delta}^{0c}(s), \widetilde{\delta}^{0a}(s), \\ & \widetilde{\delta}^{1r}(s), \widetilde{\delta}^{1d}(s), \widetilde{\delta}^{1c}(s), \widetilde{\delta}^{1a}(s)\} \end{aligned} \tag{1.5}$$

which are used in the propagation phase to perform the search.

Finally, to take full advantage of PatchMatch's efficiency, the post-processing phase must be equally fast. With this aim, an *ad hoc* post-processing was implemented, called dense linear fitting (DLF). An affine model is fit locally to each point of the offset field, with parameters estimated from the data themselves. The fitting is typically good in correspondence of a copy-moved regions, where the offset field is either constant (for plain copy-moves) or linearly varying (in the presence of rotations or resizing). On the

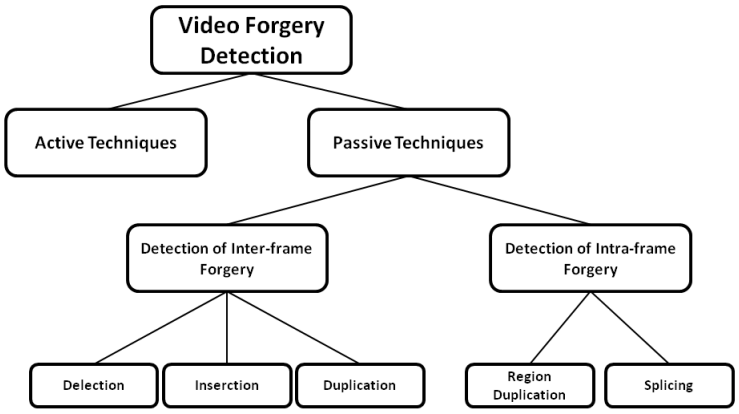


Figure 1.20: Video forgery detection: classification scheme

contrary, in pristine areas of the image, with a more chaotic field, a worse fit is typically observed. Therefore, by looking for large areas with low fitting error, copy-moves can be reliably detected. The fitting procedure is very fast, as it only requires a few linear filtering and products per pixel.

1.4 Overview on Video Copy-Move Forgery Detection

The large-scale manipulation of videos can be considered a quite recent activity. Nonetheless, several review paper have been already published on this topic [91, 95, 96]. In [96] a simple classification of methods is proposed, shown in Fig.1.20

There is a first distinction between active and passive techniques. The former, based for example on watermarking, are outside the scope of this work and neglected in the following, as already done for images. Passive techniques are further divided in inter-frame and intra-frame. The former try to discover anomalies induced in the temporal structure of the encoded stream [98], or other types of inconsistency, like artifacts due to double encoding [110, 44], and irregularities in motion-compensated edges [99] or in the velocity field [115]. To detect whether a group of frames has been deleted the use of *ad hoc* statistical features extracted from the motion residual has also been proposed [39, 86].

In recent years, only a few pioneering papers have addressed the detection of video object, that is, intra-frame, forgeries. Coding-based methods have

been proposed in [112, 101, 61, 49] where artifacts introduced by doubly-compressed MPEG videos are used as evidence of tampering. An alternative camera-based approach relies on detecting the camera "fingerprint" (camera PRNU pattern) as already done for images [11, 16]. In [74] the camcorder fingerprint is estimated on the first frames of the video and used to detect various types of attacks. A similar idea is followed in [50, 12, 36, 77] where manipulations are discovered by extracting and analyzing some suitable features from the noise residues of consecutive frames. In [55], instead, the camera-dependent photon shot noise is used as an alternative to the PRNU for static scenes.

In the classification tree, special attention is devoted to copy-move forgeries (duplication) both inter- and intra-frame, emphasizing the importance of this type of attack in the video case. Generally, techniques that implement algorithms of copy-move detection make use of *similarities or correlations* calculation between the video frames regions. The technique showed in [50] calculate the residual error in each video frame and evaluate the correlation of this residual along the temporal direction. If there is a tampered region, the calculated correlation is subjected to a more significant change than other video regions. This technique is suitable for static videos and frame-replication detection. The authors in [14, 45] calculate the residual parameters on video's patches and evaluate the correlation between the adjacent patches using the same hypothesis of [50]. Anomalies in the values of correlation show the presence of a forgery. Even this technique is only applicable to static videos (no camera-motion). In [34] authors state that the presence of a forgery provokes an unnatural similarity among the frames' blocks couples of the video. This technique can be applied only to inter-frame forgeries, in addition it is not applicable in a case of compression. In [68] a connection measure between the consecutive frames is computed and irregularities are searched for the considered time-space consistency measure. These irregularities reveal the presence of the forgery. In [81] a detection technique is presented for the objects' removal. This technique uses Scale Invariant Feature Trasform (SIFT) and k-NN search for the detection of spatial copy-move. On the contrary, for the detection of temporal copy-move it makes the video frames residual noise cross-correlation calculation. This algorithm is applicable only if the forgery copy-move is spatial or temporal. In [21] a technique is presented for the detection of copy-moves produced by removing movable objects from the video. The article is based on the hypothesis that the removal of these objects causes the insertion of artifacts in the video sequence. This technique is applicable

only to videos with a stationary background. In [9] authors use optical-flow anomalies for the video forgery detection. This technique assumes a priori knowledge of the suspected regions in the video, and can be applied only to MPEG compressed videos. In [111] the correlation coefficient is used as a measure of similarity for detecting large copy-moved blocks, while [7] extends the same method to use spatio-temporal blocks. However, this approach works well only if the cloned area is relatively large and has not been subject to subsequent post-processing. Performance drops in the presence of compression, blurring, geometric transformations and change of intensity. This is not immaterial, since these operations are often needed to make the forgery more realistic, and can be even enacted on purpose by a skilled forger to fool forensic tools.

For what concerns localization, [66] addresses only the case of whole frames inserted in the video. [100], instead, proposes a method based on HOG features and exhaustive search, which is more general but so computation-intensive to be inapplicable in practice. It should be realized that a carefully crafted copy-move may be very hard to discover by means of statistical approaches, because the copied object has the same statistics as the background, unless it is rotated or resized. In addition, occlusive copy-moves, based on the copy of background areas, do not offer visual clues or salient keypoints (see again Fig.1.15) which enable their discovery. Finally, a clever attacker may enact further expedients to confuse matching-based methods, like temporally flipping the inserted video object.

1.4.1 A Method for Detection of Block Duplications in Videos

In this section we focus on the video copy-move forgery detection algorithm proposed in [7]. So far, this technique was the only method proposed in literature working under the hypothesis mentioned at beginning of this chapter; i.e., as the technique proposed in this thesis, the algorithm proposed in [7]: *i)* is designed to detect region duplication, where the source of the cloned region belongs to video under test itself, and *ii)* is completely blind with respect to the video format. For this reason, in chapter 5, we will compare the performance of the proposed algorithm with this technique. The algorithm is basically a correlation method able to detect copy-move forgeries and, in the case a forgery is detected, it can localize the temporal position of the tampered region, i.e. it reveals the video's tampered frames, but not its precise spatial position.

The idea is to detect duplicated regions by cross-correlating small 3D blocks of the video residual. Let V be the video under test whit size $I \times J \times T$,

firstly the video residual R_V is calculated by computing difference between adjacent frames; let $x^t_{i,j}$ be the gray value of the pixel of the video V, and $r^t_{i,j}$ the value of the video residual R_V in position (i, j, t), with $(i, j, t) \in [1, I] \times [1, J] \times [1, T-1]$:

$$r^t_{i,j} = x^t_{i,j} - x^{t+1}_{i,j} \tag{1.6}$$

In order to reduce the computational complexity, a $5\times$ downsampling is carried out in the spatial domain, while no downsampling is applied in the time domain to retain the full temporal resolution. In the second step, the resized version of R_V (let it be R) is split into non overlapping 3D blocks B^n_m of size $\mathrm{d}i \times \mathrm{d}j \times \mathrm{d}t$, where n is the starting time index of B^n_m and $m \in [1, M]$ (M is the total number of blocks). The next step of the algorithm consists of computing the phase-correlation between B^n_m and R as:

$$C^t_{i,j}(B^n_m) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(B^n_m)\mathcal{F}(R)^*}{|\mathcal{F}(B^n_m)\mathcal{F}(R)^*|}\right) \tag{1.7}$$

where $\mathcal{F}$ is the Fourier transform operator, and * indicates the complex conjugate. This phase-correlation is computed for each block B^n_m. The quantity $C^t_{i,j}(B^n_m)$ denote the similarity between the block B^n_m and R. After this step, the maximum correlation value $c^t_{B^n_m}$ for each position time is computed as:

$$c^t_{B^n_m} = \max_{i,j}\left(|C^t_{i,j}(B^n_m)|\right) \tag{1.8}$$

In Fig.1.21 the behavior of $c^t_{B^n_m}$ is shown for a duplicated and a non-duplicated block.

Each block has a peak due to autocorrelation. Only if the block belongs to a duplicated region, a second prominent peak should appear in $c^t_{B^n_m}$. So, if the second peak is sufficiently large (the authors suggest at least 0.6 times the first peak), a confidence value is associate to the block B^n_m as:

$$p_{B^n_m} = \frac{\max\left(c^t_{B^n_m}\right)}{\frac{1}{T-1}\sum_t c^t_{B^n_m}} \tag{1.9}$$

The value $p_{B^n_m}$ is computed for each block, and the block corresponding to the largest of all $p_{B^n_m}$ is the most likely duplicate candidate. For the final decision, this block is compared with those starting from the frame in time position $\tilde{n}$ where $\tilde{n}$ is the index where the second peak of $c^t_{B^n_m}$ is located. Let this block

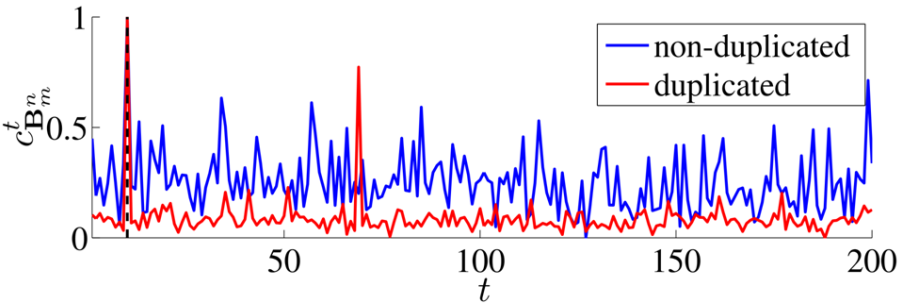
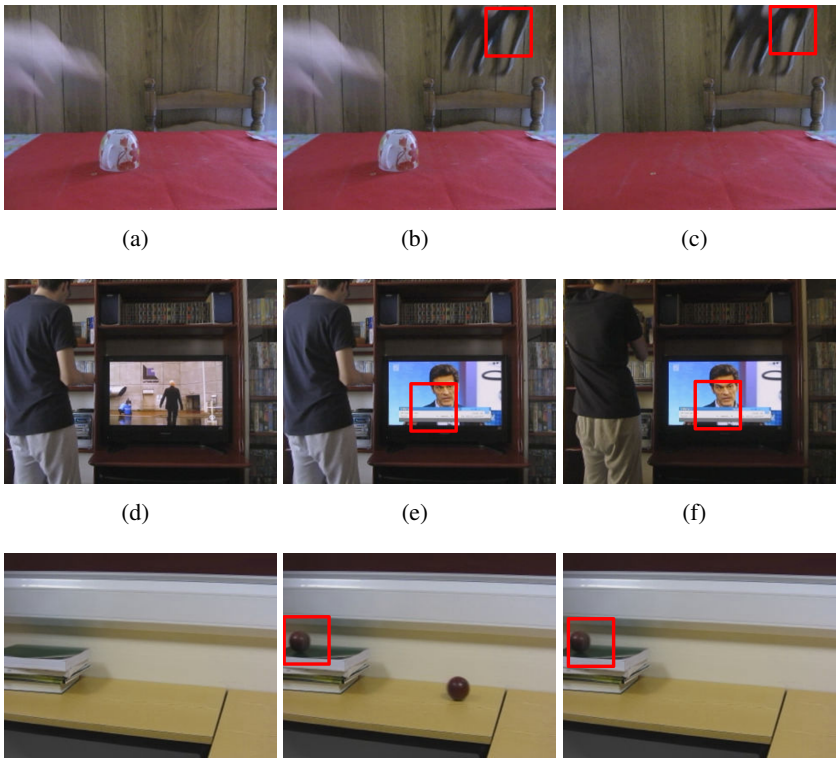


Figure 1.21: Trends of $c^t_{B^n_m}$ for a nonduplicated block (blue), and for a duplicated one (red). The dashed line represent the starting time of B^n_m. In the red line a second prominent peak appears, that indicates the temporal position of the clone.

be $B^{\tilde{n}}_{\tilde{m}}$, a forgery is detected if a matching block is detected, i.e. the following condition occurs:

$$\text{MSE}\left(B^n_m, B^{\tilde{n}}_{\tilde{m}}\right) < \tau \tag{1.10}$$

where MSE is the mean square error between two blocks and τ is a threshold defined by user to tune the tradeoff between false positives and false negatives. In Fig.1.22 an example of detection performed by the algorithm just described is shown.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 1.22: Examples of block duplication detection for 3 sequences. Original (left), forged (middle), and frames with duplicated blocks (right). The red shape highlights a detected duplication.

Chapter 2

Feature Extraction

As already said, to perform efficiently and effectively the search of clones through the image or video it is necessary to associate a compact and expressive feature with all (or some) target pixels, computed on suitable patches surrounding them. These features should possess several invariance properties in order to be robust with respect to a number of possible sources of distortion and deformation.

Suppose to have a forged video V where a given object with support Ω_1 has been copy-moved to $\Omega_2 = T(\Omega_1)$, with $T(\cdot)$ an affine transform (indicating for example a translation and/or rotation of the object.) Let us now consider patch P_1 anchored to pixel $s_1 \in \Omega_1$; under the affine transform it will be mapped to another patch $P_2 = T(P_1)$ associated with pixel $s_2 = T(s_1) \in \Omega_2$. Due to geometric distortions induced by $T(\cdot)$, the two patches will not be identical, and they may even differ significantly under naive distortion measures, like the mean square error. Just think of the case in which P_2 is rotated by 180 degrees with respect to P_1. In order for the matching algorithm to discover that these patches are indeed related, we need to associate with each patch a feature $\boldsymbol{f} = \Psi(P)$ which is invariant to translation and rotation. Moreover, if the detection algorithm must be robust with respect to other distortions, like compression, or noise addition, the transform $\Psi(\cdot)$ should be also invariant with respect to small differences between the patches.

More formally, let $\boldsymbol{f}_1 = \Psi(P_1)$ be the feature vector associated with patch P_1 and $\boldsymbol{f}_2 = \Psi(P_2)$ the feature vector associated with patch $P_2 = T(P_1)$, we want a transform $\Psi(\cdot)$ such to minimize the Euclidean distance between $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$, irrespective of the affine transform $T(\cdot)$. On the contrary, $\Psi(\cdot)$ should make the distance between features associated with patches that are unrelated under

$T(\cdot)$ relatively large, that is, it should have a high discriminative power.

Invariant features have been the object of intense research for years, especially in Computer Vision, and a large number of them have been proposed which possess a number of interesting properties. Indeed, many of these features have been already used for forensic applications [67, 6, 88, 20, 89, 64]. Here, we mention only some of the most powerful and popular, like *Scale Invariant Feature Transform* (SIFT) [70], *Speeded Up Robust Feature* (SURF) [5], *Fast Local Descriptor for Dense Matching* (DAISY) [106], *Shift Invariant Descriptor* (SID) [56], *Histogram of Oriented Gradients* (HOG) [31], *Local Binary Patterns* (LBP) [78], *Zernike Moments* [119]. However, we will not go into further details, except for the Zernike features used in this work, referring the reader to the original papers for a deeper analysis.

In the context of this thesis, it is important to focus on whether the features are extracted densely in the image or else are associated with selected keypoints. A keypoint is a pixel associated with some salient characteristic of the image, like for example a corner, or more in general a point that could raise the interest of a human observer and be easily found by a specialized detector. As an example, SIFT features are usually computed on keypoints that are selected as maxima in a suitable space, and are typically characterized by the presence of significant image gradients. Therefore SIFT features are not extracted in homogeneous regions of an image. On the other hand, SIFT features guarantee rotation invariance thanks to the estimation and use of the dominant direction in the patch: for homogeneous patches, a dominant direction may not exist or be unreliably estimated, due for example to added noise or compression, providing unreliable information for subsequent uses.

Therefore, only some features can be meaningfully computed densely on the image, irrespective of the patch nature. This property will be important when addressing the problem of occlusive copy-moves. In fact, when a homogeneous region is copy-moved to hide an object it will hardly give rise to significant keypoints. Therefore, to find such kind of attacks, features must be computed densely on the image and should be able to provide valuable clues also on smooth regions.

In the following we will focus on a specific class of features, the Zernike moments, belonging to the family of Circular Harmonic Transforms (CHT), which possess desirable invariance properties and keep providing valuable information even when computed on smooth regions. Moreover, the Zernike moments have already been used with success in copy-move forgery detection [89]. The following description is inspired to [23].

2.1 Zernike Moments

In this section we describe the process that leads us to definition of the features used in our detection algorithm. These features were originally designed for still images. In fact, calculation of Zernike Moments for each frame of the video is the first step to compute our final features. As already said, Zernike Moments belong to the family of Circular Harmonic Transforms (CHT), that we are going to present now.

Let us consider a scalar image $I(x, y)$ defined on a continuous space, $(x, y) \in R^2$. We can represent the image in polar coordinates as $I(\rho, \theta)$, where $\rho \in [0, \infty]$ and $\theta \in [0, 2\pi]$.

To define the CHT we consider the basis functions $K_{n,m}(\rho, \theta)$ defined as

$$K_{n,m}(\rho, \theta) = R_{n,m}(\rho)\frac{1}{\sqrt{2\pi}}e^{jm\theta} \tag{2.1}$$

where $R_{n,m}(\rho)$ is called radial profile and $\frac{1}{\sqrt{2\pi}}e^{jm\theta}$ is a circular harmonic. The image's transform is

$$\boldsymbol{f}_I(n, m) = \int_0^{2\pi}\int_0^{\infty} I(\rho, \theta)\, K^*_{n,m}(\rho, \theta)\rho \,\mathrm{d}\rho \,\mathrm{d}\theta \tag{2.2}$$

We can rewrite the previous equation as

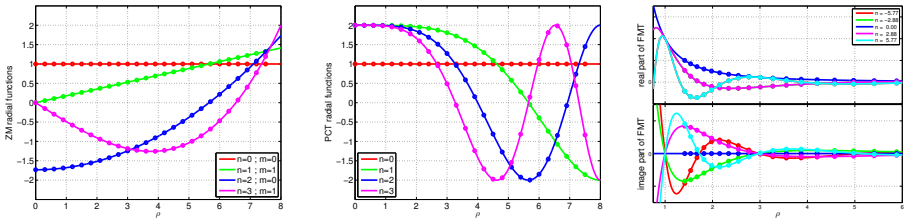
$$\boldsymbol{f}_I(n, m) = \int_0^{\infty} \rho R^*_{n,m}(\rho) \times \left[\frac{1}{\sqrt{2\pi}}\int_0^{2\pi} I(\rho, \theta)e^{-jm\theta}\mathrm{d}\theta\right] \mathrm{d}\rho \tag{2.3}$$

The integral in square brackets, let it be $\widehat{I}(\rho)$, is the Fourier series of $I(\rho, \theta)$ along the angle coordinate. This results explains CHT rotation invariance. In fact, a rotation of θ_0 radians in I can be written in polar coordinates as an operation that turns $I(\rho, \theta)$ into $I(\rho, \theta + \theta_0)$. As well known, this contributes just a phase term $e^{jm\theta_0}$ in the Fourier Transform $\widehat{I}$, which can be estimated and compensate or simply removed by taking the magnitude of the coefficients.

The various CHTs differ in the radial profile. Various choices have been considered in literature for the most appropriate radial profile. For example, in [117] authors consider Polar Cosine Transform (PCT), where the radial profile is defined as

$$R_n(\rho) = C_n \cos(\pi n \rho^2) \tag{2.4}$$

where $\rho \in [0, 1]$ and C_n are normalizing coefficients.



(a) (b) (c)

Figure 2.1: Radial profiles of some CHTs: Zernike (a), PCT (b), FMT (c).

In [93] the Fourier-Mellin Transform (FMT) is chosen, where the radial profile is

$$R_\nu(\rho) = \frac{1}{\rho^2} e^{j\nu \ln(\rho)} \tag{2.5}$$

where $\rho \geq 0$ and ν is a continuous parameter. With this choice, the integral in eq.2.3, after a coordinate mapping, can be regarded as the bi-dimensional Fourier transform of I in log-polar coordinates. This guarantees scale invariance properties to the transform.

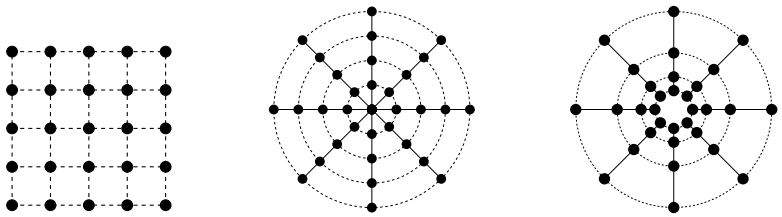
For our algorithm we chose the Zernike Moments (ZM), already used in [103] for image analysis. In this case the radial profile is

$$R_{n,m}(\rho) = \sum_{h=0}^{(n-|m|)/2} C_{n,m,h} \rho^{n-2h} \tag{2.6}$$

where $\rho \in [0, 1]$ and $C_{n,m,h}$ are suitable coefficients that ensure orthonormality of the basis functions. Examples of these radial profiles are shown in Fig.2.1.

In several applications, including forensics, Zernike Moments have proven quite robust with respect to small distortions of the image, like noise addition, moderate compression, etc., which is the main reason that led us to prefer it to other solutions.

The problem discussed so far is defined in a continuous domain. In the rest of the paragraph we explain how to translate these theoretical definitions into practical formulas, since we need to compute our features on a discrete grid (the digital image), preserving the invariance properties. As a first step, we have to choose a patch size, since computation must regard only a limited support for the sake of efficiency. This parameter affects both discrimination and robustness. In fact patches too small might not catch the local image behavior,



(a) (b) (c)

Figure 2.2: Examples of rectangular (a), polar (b) and log-polar (c) sampling grids.

while patches too large might loose resolution. Then, we have to choose a suitable finite number of (n, m) couples, not too large, to limit computational cost and also memory storage, but neither too small, to avoid losing discrimination ability. Finally, we must approximate the integral (2.2) with a summation over the patch.

There are two ways to do the approximation. The first way consists in resampling the basis functions $K_{n,m}(\rho, \theta)$ on the grid points (x, y) of the patch. Let W be the domain where the path is defined, the integral in eq.2.2 becomes

$$\boldsymbol{f}_I^{'}(n,m) = \sum_{(x,y)\in W} I(x,y)\, K^*_{n,m}(\rho(x,y), \theta(x,y)) \tag{2.7}$$

with $\rho(x, y) = \sqrt{x^2 + y^2}$ and $\theta(x, y) = \pm \arctan(y/x)$.

Otherwise, we can resample the image on polar (or logpolar) coordinates (see Fig.2.2(b)-(c)), In this case eq.2.2 becomes

$$\boldsymbol{f}_I^{''}(n,m) = \sum_{(\rho,\theta)\in W} I(x(\rho,\theta), y(\rho,\theta))\, K^*_{n,m}(\rho,\theta)\rho^i \tag{2.8}$$

with $i = 1$ for the polar grid and $i = 2$ for the logpolar one. These two ways to perform the approximation of the integral entails non-negligible consequences on performance, as shown in [116]. In particular, polar sampling guarantees perfect invariance for rotation angles multiple of the sampling step $\Delta\theta$. Instead, features computed in the first way (on the cartesian grid) can entail the loss of the rotation invariance, mainly for angles close to $\pi/4 \pm k\pi/2$ [65]. For these reasons we chose the polar sampling to compute the features for our experiments. Moreover, after some preliminary experiments, we chose a patch size of 16×16 and kept only 12 moments, that is, 12 (n, m) couples.

2.2 3D Flip-Invariant Features

Building upon the still-image copy-move detector of [26] we begin by associating with each pixel a feature vector composed by the Zernike moments computed on a polar grid centered on the target. Dealing with a video source, however, we have the opportunity to extract features from 3D rather than 2D patches. With this choice a more expressive feature is obtained, accounting also for informative temporal changes. On the down side, 3D patches may be less effective with forgeries of very short duration or with fast moving objects, because many 3D patches would include both pristine and copy-moved regions, making it difficult to find the correct match. Moreover, 3D features are more fragile with respect to temporal distortions, such as flipping or temporal down/up-sampling which may be used to modify the speed of an object. Since both solutions (3D-patch or 2D-patch based features) have pros and cons, we will test them both in the experiments. For the second case, however, we define a flip-invariant feature so as to be robust to the simplest form of tampering in the temporal dimension.

Specifically, let $\boldsymbol{f}(s,t,n,m)$ be a generic feature associated with the t-th frame of the video, for spatial location s and Zernike moment (n,m). Then, the 2D-patch feature vectors used in the algorithm, or 2D features for short, will be defined as

$$\mathbf{f}^{\mathrm{2D}}(s,t) = \{\boldsymbol{f}(s,t,n,m),\ (n,m) \in \mathcal{F}^{\mathrm{2D}}\} \tag{2.9}$$

where $\mathcal{F}^{\mathrm{2D}}$ identifies a subset of all Zernike moments[1].

This subset should be as small as possible to limit complexity and memory problems in the algorithm, while including sufficient discriminative information on the patch.

Now, we could define 3D-features as

$$\mathbf{f}^{\mathrm{3D}}(s,t) = \{\boldsymbol{f}(s,t+\tau,n,m),\ |\tau| \leq T, (n,m) \in \mathcal{F}^{\mathrm{3D}}\} \tag{2.10}$$

where Zernike moments from $2T$+1 consecutive frames are taken, and $\mathcal{F}^{\mathrm{3D}}$ identifies a new subset of Zernike moments[2], smaller than before to limit the overall feature size when $2T$+1 such moments are stacked. This latter feature, however, is not flip invariant, and would not allow the detection of clones

[1] $(n,m) \in \{\ (0,0); (1,1); (0,2); (2,2); (1,3); (3,3); (0,4); (2,4); (4,4); (1,5); (3,5); (5,5)\}$

[2] $(n,m) \in \{\ (0,0); (1,1); (0,2); (2,2); (1,3); (3,3)\}$

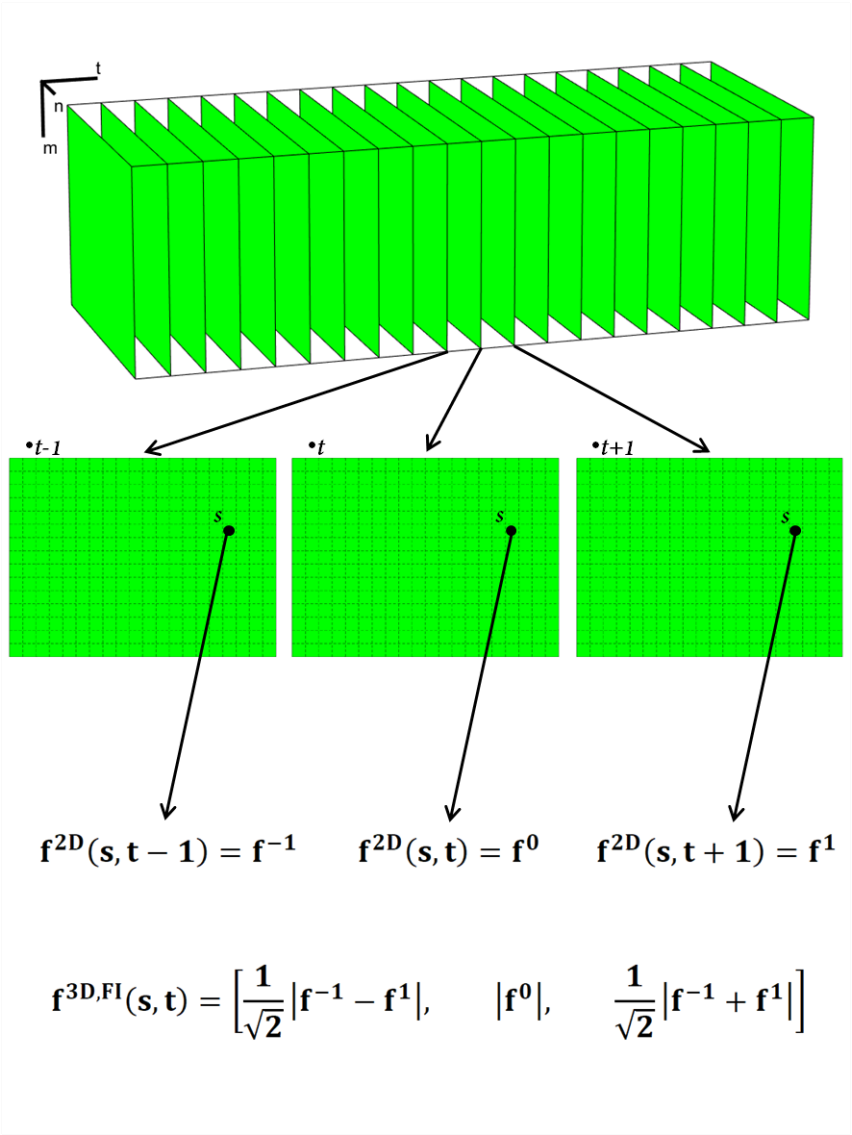


Figure 2.3: Example of features 3D computation (with $T = 3$ in Eq.2.10). From top to bottom: 2D features of all video sequence; 2D features from frames $t-1$, t and $t+1$; 2D features from frame $t-1$, t and $t+1$ in position s; 3D feature in position (s,t);

played backwards in time. Therefore, to improve robustness to malicious attacks, we modify it as

$$\mathbf{f}^{\mathrm{3D,FI}}(s,t) = \{\, \boldsymbol{g}(s, t{+}\tau, n, m),\ |\tau| \leq T, (n,m) \in \mathcal{F}^{\mathrm{3D}}\} \tag{2.11}$$

where, displaying only the dependence on the temporal index,

$$\boldsymbol{g}(t{+}\tau) = \begin{cases} \frac{1}{\sqrt{2}}|\boldsymbol{f}(t{+}\tau) + \boldsymbol{f}(t{-}\tau)| & \tau > 0 \\ |\boldsymbol{f}(t)| & \tau = 0 \\ \frac{1}{\sqrt{2}}|\boldsymbol{f}(t{+}\tau) - \boldsymbol{f}(t{-}\tau)| & \tau < 0 \end{cases} \tag{2.12}$$

The even-odd transform of Eq.(2.12) guarantees the desired flip invariance. The scheme of this transformation is shown in Fig.2.3.

Note that the transform is applied on the Zernike moments, not their absolute values. By so doing, the $\mathbf{f}^{\mathrm{3D,FI}}$ vectors are invariant to spatial rotations of the whole 3D patch. Taking the absolute value of the moments before the even-odd transform would enforce invariance with respect to different rotations for each frame. This is a useless property for our problem, since temporal and spatial manipulations do not interfere with one another. We also note explicitly that the above formulas refer to a 3D patch with an odd number of frames, similar formulas apply for the even number case.

Chapter 3

Matching Algorithm

Once all features have been extracted, and they represent faithfully the patches they are associated with, it is necessary to find for each of them the best matching feature, or nearest neighbor. When a copy-move is present in the image or video, all pixels belonging to a clone exhibit coherent nearest neighbors, namely, with the same spatial and temporal offset from the source. Therefore, the problem decomposes to *i)* finding the nearest neighbor field (NNF) and *ii)* processing this field to decide whether a region with coherent offsets exist and is significant. In this chapter with deal with the first problem, matching.

Finding the nearest neighbor to a given point in a set of D-dimensional points is a fundamental problem in signal processing and computer vision, with uncountable applications, from vector quantization to image retrieval, to name a few. Exhaustive search becomes very quickly unfeasible as the cardinality of the dataset grows beyond a few thousands. Dealing with images, the number of features to match is in the order of millions and grows towards the billions when considering videos. Therefore, it is mandatory to consider some fast methods for approximate nearest neighbor (ANN) search. In the last decades there has been intense research on this topic, and a large number of ANN search algorithms have been proposed. Among the most popular and effective, we mention those based on locality sensitive hashing (LSH) [35, 109, 107], on k-D trees [94, 76], and on product quantization [51, 2], referring the interest reader to the original papers for more information. Unfortunately, although extremely effective, these algorithms can reduce the complexity (keeping a good accuracy) by only two-three orders of magnitude with respect to exhaustive search [107], which is not enough to meet the complexity challenges offered by dense field matching in images and videos.

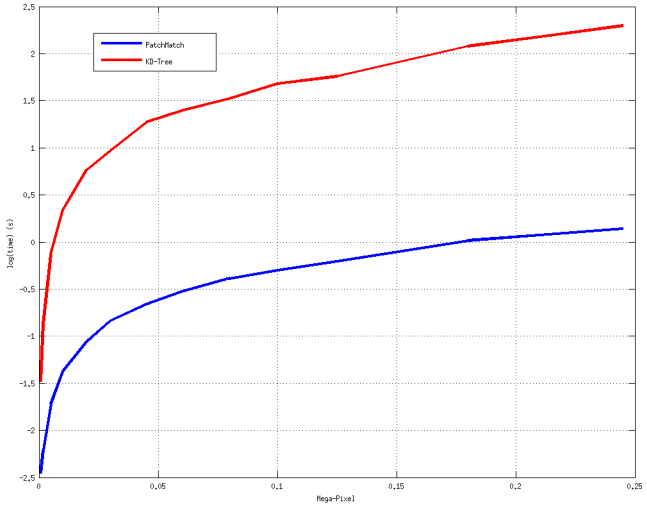


Figure 3.1: Comparison between CPU time (in log-scale) of PatchMatch (blue) and Kd-Tree (red) as a function of the size of the data structure (Mpixel). Time for kd-tree includes tree creation. For PatchMatch 10 iterations are considered.

However, these algorithms have been developed for generic sources and do not take into account the peculiar nature of our problem, finding a spatial nearest neighbor field associated to a typically smooth image or video. Thanks to this further information, one can assume that the NNF itself is generally smooth, which provides a formidable asset to reduce complexity. In particular, this hypothesis is exploited in the PatchMatch algorithm, proposed originally [3] in 2009, which indeed reduces the search complexity hugely, allowing to solve our matching problem with reasonable complexity and very good accuracy. As an example Fig.3.1 shows the trends of the computation time for KD-tree and PatchMatch w.r.t. number of pixel of the searching field.

In fact, PatchMatch exploits the intrinsic smoothness of multimedia sources assuming that if a matching exist between features at pixel s_1 and s_2, it is very likely that feature at pixel $s_1 + \Delta_s$ matches a feature at pixel $s_2 + \Delta_s$ or very close to that. This property, which holds very often in images and video, is the key to PatchMatch speed. In the following, we describe PatchMatch and its proposed modifications in more detail.

3.1 PatchMatch

Let $I = \{I(s) \in R^K, s \in \Omega\}$ be an image defined over a regular rectangular grid Ω. With each pixel s we associate a feature vector, $\boldsymbol{f}(s)$, which describes the image patch centered on s. Given a suitable measure of distance between features, $d(\boldsymbol{f}', \boldsymbol{f}'')$, we define the nearest neighbor of s as the pixel, $s' \in \Omega$, which minimizes the feature distance w.r.t. s over the whole image

$$\mathrm{NN}(s) = \arg\min_{s' \in \Omega} d(\boldsymbol{f}(s), \boldsymbol{f}(s')) \tag{3.1}$$

Rather than the nearest-neighbor field (NNF) itself, in the following we will consider the equivalent offset field, with the offset defined as $\delta(s) = \mathrm{NN}(s) - s$.

PatchMatch is a randomized iterative algorithm for NNF estimation. As all iterative algorithms, convergence to the desired solution is much faster in the presence of a good initial guess. With images, however, such a good guess is easily obtained, because their NNFs are typically constant or linearly varying over large areas, as a consequence of image smoothness, and hence highly predictable. Given this core idea, PatchMatch is easily understood. Following a random initialization, the two phases of offset prediction and random search alternate until convergence.

Initialization. The offset field is initialized at random, as $\delta(s) = U(s) - s$. where $U(s)$ is a bi-dimensional random variable, uniform over the image support Ω. In copy-move search, we enforce an additional constraint on matches, which must be reasonably far from the target, excluding offsets smaller than a given threshold. Most of the initial offsets are useless, but a certain number will be optimal or near-optimal. These are quickly diffused to the rest of the image in the propagation phase.

Propagation. In this step, the image is raster scanned top-down and left-to-right (with scanning order reversed at every other iteration), and for each pixel s the current offset is updated as

$$\delta(s) = \arg\min_{\phi \in \Delta^P(s)} d(\boldsymbol{f}(s), \boldsymbol{f}(s + \phi)) \tag{3.2}$$

where $\Delta^P(s) = \{\delta(s), \delta(s^r), \delta(s^c)\}$, and s^r and s^c are the pixels preceding s, in the scanning order, along rows and columns, respectively. Therefore, the algorithm uses the offset of nearby pixels as alternative estimates of the current offset, and selects the best one. If a good offset is available for a given pixel

of a region with constant offset, this will very quickly propagate to the whole region.

Random search. To avoid getting trapped in bad local minima, after each propagation step a random search step follows, based on a random sampling of the current offset field. The candidate offsets $\delta_i(s), i = 1, \ldots, L$ are chosen as $\delta_i(s) = \delta(s) + R_i$ where R_i is a bi-dimensional random variable, uniform over a square grid of radius 2^{i-1}, excluding the origin. In practice, most of these new candidates are pretty close to $\delta(s)$, but large differences are also allowed, with small probability. Given the rare sampling, only a few new candidates are eventually selected. The random-search updating reads therefore as

$$\delta(s) = \arg \min_{\phi \in \Delta^R(s)} d(\boldsymbol{f}(s), \boldsymbol{f}(s + \phi)) \tag{3.3}$$

where $\Delta^R(s) = \{\delta(s), \delta_1(s), \ldots, \delta_L(s)\}$.

Experiments [3] show that typically PatchMatch converges to a near-optimal NNF in less than 10 iterations.

3.2 Modified PatchMatch

Given the very good performance of PatchMatch, several research teams have soon proposed modified versions of the basic algorithm. In [58] and in [79], fast approximate NN search techniques are used in place of the random initialization, possibly with spatial sub-sampling, reducing drastically the number of iterations required for convergence, and hence the overall processing time. The authors of PatchMatch proposed its generalized version [4], which can deal with rotated and rescaled objects by working in a 4D space including scale and angle besides the spatial coordinates. Unfortunately, this version can work only on image samples (not compact features), and is significantly more complex than the basic version due to the need of patch resampling and the slower convergence.

A much simpler, yet effective, modification is proposed in [22] which involves only the propagation step, where better offset predictors are included. The main observation is that basic PatchMatch uses only zero-order predictors in the propagation phase, which makes sense when the offset field is constant (e.g. rigid copy-move forgeries) but not in the presence of rescaling and rotation, in which cases a linearly varying offset field is obtained. Therefore, in [22] the set of predictors is enlarged to include zero-order and first-order

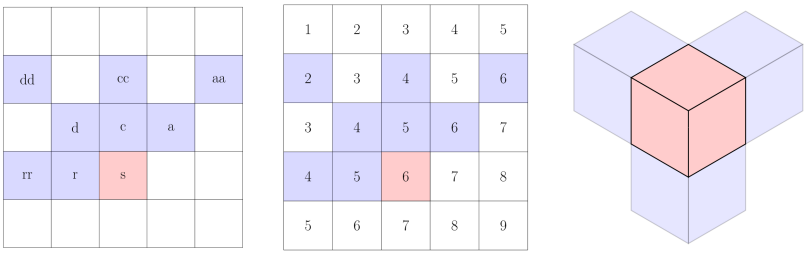


Figure 3.2: Left: pixels used to compute the predictors of $\delta(s)$. Center: with a linearly varying offset field, zero-order predictors may be wrong: $\hat{\delta}^{0r}(s) = \delta(r) \neq \delta(s)$, while first-order predictors are always correct: $\hat{\delta}^{1r} = 2\delta(r) - \delta(rr) = \delta(s)$. Right: with videos, predictors develop also along the third (frame f) direction.

predictors along several directions (row, column, diagonal antidiagonal, see Fig.3.2, that is

$$\begin{aligned} \Delta^P(s) \quad = \quad & \{\hat{\delta}(s), \hat{\delta}^{0r}(s), \hat{\delta}^{0d}(s), \hat{\delta}^{0c}(s), \hat{\delta}^{0a}(s), \\ & \hat{\delta}^{1r}(s), \hat{\delta}^{1d}(s), \hat{\delta}^{1c}(s), \hat{\delta}^{1a}(s)\} \end{aligned} \tag{3.4}$$

where $\hat{\delta}^{ix}(s)$ indicates the i-th order predictor along direction x, and hence $\hat{\delta}^{0x}(s) = \delta(x)$ and $\hat{\delta}^{1x}(s) = \delta(x) + [\delta(x) - \delta(xx)]$.

With this modification, linear offset fields can be easily tracked along the image, with a negligible increase in complexity. Moreover, the algorithm can make use of any features, including scale- and rotation-invariant ones, which results in improved localization of copy-move forgeries [22, 26].

3.2.1 Adapting PatchMatch to video

To take advantage of the features' invariance to rotations, we use the modified version of PatchMatch proposed in [26]. Then, to deal with a video source, we adopt some further straightforward modifications, originally developed in [32]. First of all, while keeping the general structure of the algorithm, we include further predictors to take into account the temporal direction, and in particular the zero-order and first-order predictors along frames $\widehat{\delta}^{0f}(s)$ and $\widehat{\delta}^{1f}(s)$. Using first-order prediction along time allows one to deal also with subsampling [upsampling] in the temporal direction, corresponding to a change of speed in moving objects. In addition, the random search step is also modified to sample the whole 3D space, testing offsets taken at random in the whole datacube.

Despite the increased size of the source, with a large number of frames, the same number of candidates is used as for still images. Since this procedure is repeated for all pixels in the video, a large number of near-optimal offsets are sampled anyway, and then propagated to the whole video.

3.3 PatchMatch with Multiresolution

The overall complexity of the image-based algorithm [26] is clearly dominated by the matching phase, accounting for about 75% of the total CPU-time, with the computation of features taking another 15%, and the post-processing phase the remaining 10%. In the case of video, these proportions do not change much. Therefore all efforts should be devoted to further reduce the cost of computing the NN field.

Using PatchMatch, the complexity of computing the NN field, measured in number of multiplications, can be approximated as

$$\mathrm{c} = \mathrm{n_{it}}\mathrm{n}(\mathrm{c_P} + \mathrm{c_R})\mathrm{f} \tag{3.5}$$

where $\mathrm{n_{it}}$ is the number of iterations of PatchMatch, n the number of pixels in the video, $\mathrm{c_P}$ and $\mathrm{c_R}$ are the number of candidate offset tested in the propagation and random search phases, respectively, and f is the feature length. n, in its turn, is the product of frame size and number of frames. Using typical values, that is, frames of 0.5 Mpixels, 8 iterations of PatchMatch, with 10 candidates tested in each phase, and features of length 10, the overall complexity is 8×10^8 multiplications per frame. At 25 frames/second, this represents a huge computational burden, even for short videos.

The only effective way to reduce significantly this burden is through subsampling. Indeed, keypoint-based methods use exactly this strategy, computing matches only for some sparse keypoints. As we follow a dense-field approach, we perform instead a regular $S \times S$ subsampling (that is, we take one every S-th pixel along rows and columns). Note that features are always computed on the original frames *before* subsampling, therefore they represent patches observed at the native resolution, say level-0, with no loss of information. Also, no subsampling is performed along the temporal direction.

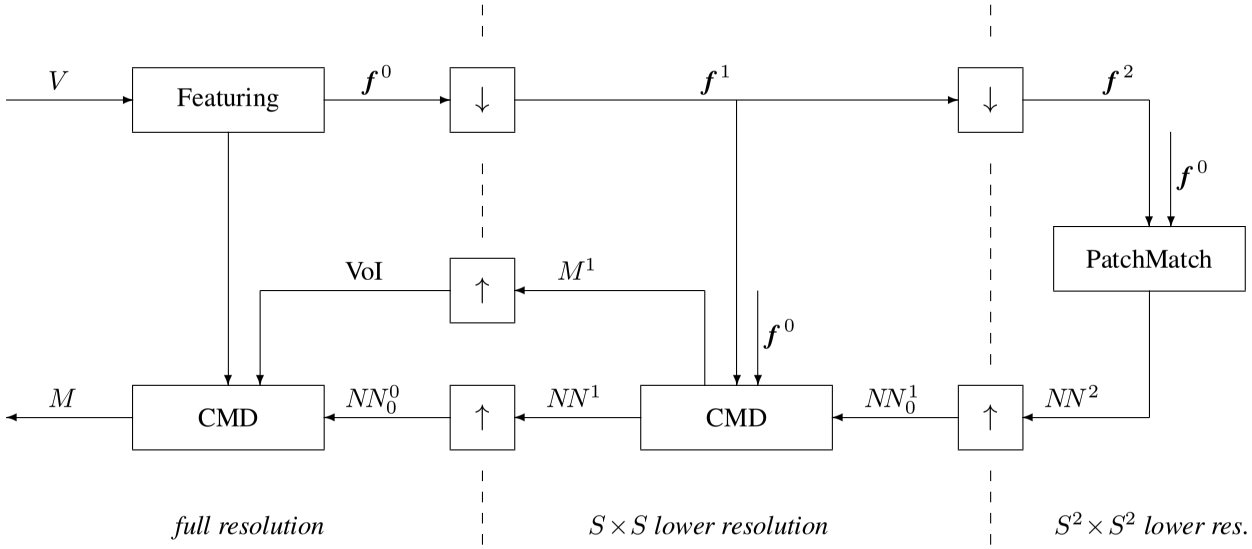


Figure 3.3: Block diagram of the proposed video copy-move detector with multi-resolution processing. The high-resolution field of features $\boldsymbol{f}^0$ is extracted from the original video, V. This field is then downsampled twice to obtain fields $\boldsymbol{f}^1$ and $\boldsymbol{f}^2$. At level 2 (lowest resolution) PatchMatch works on $\boldsymbol{f}^2$ and $\boldsymbol{f}^0$ to provide the NN field NN^2. This is upsampled to become the initial NN field at level 1, NN_0^1. At level 1, the copy-move detector (CMD) works on $\boldsymbol{f}^1$ and $\boldsymbol{f}^0$ to refine NN_0^1 to NN^1, and to extract the detection map M^1 by applying the post-processing. Copy-moved objects are detected in this level, but their shape can be recovered more precisely at level 0. So M^1 is upsampled to define the volume of interest (VoI) and NN^1 is upsampled to become the initial NN field at level 0, NN_0^0. At level 0, the copy-move detector works on $\boldsymbol{f}^0$, limited only to the VoI, to extract the final output, the detection map $M^0 = M$.

By working at level-1 resolution (that is, after subsampling) PatchMatch complexity reduces by a factor S^2, approximately, if all other parameters are kept fixed. Moreover, with a moderate subsampling step, we expect to keep detecting at level-1 most, if not all, the copy-moves detectable at level-0. Of course, there is a loss in spatial accuracy. However, this can be largely recovered by upsampling the NN field back at level-0, and running a few iterations of PatchMatch to propagate the correct offsets. Since detection has been already performed at level 1, at level 0 PatchMatch is applied only to the volumes of interest (VoI), namely the frames where copy-moves have been detected, while the random search phase is skipped altogether. With these simple modifications, the processing at level 0 does not impact heavily on the overall cost. In particular, since the regions involved in the copy-move are a little percentage of the whole video, the use of VoI alone is already very effective.

Algorithm 1 Multi-resolution Video Copy-Move Detector

Require: V ▷ input video
Ensure: M ▷ output detection map
1: $\boldsymbol{f}^0$ = FeatureExtract(V) ▷ will work on features from now on
2: $\boldsymbol{f}^1 = \boldsymbol{f}^0 \downarrow S$ ▷ $S \times S$ downsampling
3: $\boldsymbol{f}^2 = \boldsymbol{f}^1 \downarrow S$ ▷ $S \times S$ downsampling
4: NN^2 = PatchMatch($\boldsymbol{f}^2$,$\boldsymbol{f}^0$) ▷ NN field at level 2
5: $NN_0^1 = NN^2 \uparrow S$ ▷ initial estimate of NN^1
6: $[M^1, NN^1]$ = CMD($\boldsymbol{f}^1$,$\boldsymbol{f}^0$, NN_0^1) ▷ CMD at level 1
7: $M_0^0 = M^1 \uparrow S$ ▷ M_0^0 gives the VoI
8: $NN_0^0 = NN^1 \uparrow S$ ▷ initial estimate of NN^0
9: $[M, NN^0]$ = CMD($\boldsymbol{f}^0$, NN_0^0, VOI) ▷ CMD at level 0 on VoI

Therefore, the bulk of processing is now at level 1, where PatchMatch works in its standard configuration. To further reduce the processing cost we resort again to $S \times S$ subsampling, and run PatchMatch at this level-2 resolution. The retrieved NN field is then upsampled and used to initialize PatchMatch at level-1 in order to ensure its quick convergence, thus reducing the number of iterations. Note that subsampling operates only to reduce the *source* features to match, while the target features are not sampled at all, otherwise the correct offset may not be found. For example, at the lowest resolution, features drawn from F^2 are matched to features drawn from F^0. Note also that, thanks to the first-order predictors, propagation keeps working correctly. In Fig.3.3 and Algorithm 1 we show a block diagram and the pseudo code of the com-

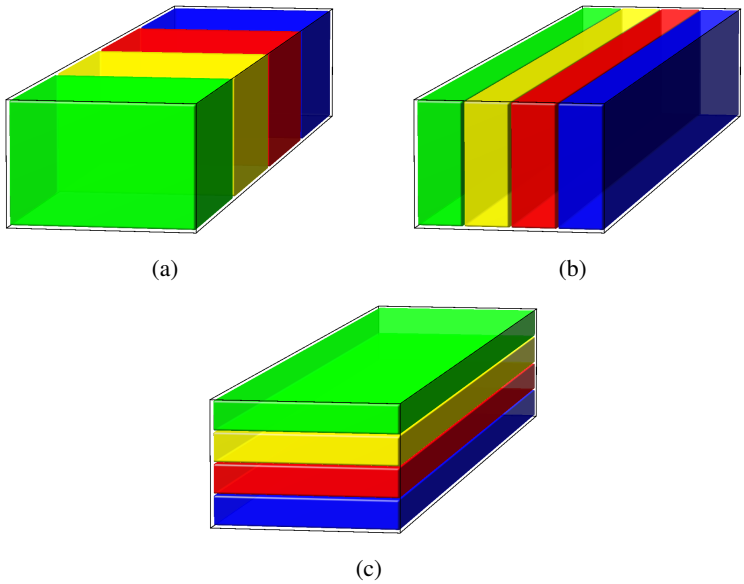


Figure 3.4: Example of video partitions for PM parallel computing with 4 threads. Each color corresponds to a region of the video assigned to a specific thread. Each two iterations of PatchMatch the configuration of video partition switchs into one of the three shown in the figure to better propagate matchs in all directions of the video.

plete multiresolution scheme.

Finally, to gain some more speed, we resorted to parallel computing. The parallel code works seamlessly for feature extraction and in the post-processing phase. As for PatchMatch, each thread operates only on a portion of the source data (while target data are not partitioned), and the offset subfields are joined after each iteration. By so doing, however, propagation of offsets across partition boundaries may be delayed significantly. Therefore, we change partitions after each couple of forward-backward iterations, orienting boundaries along columns, rows, or frames in round-robin fashion (Fig.3.4).

Chapter 4

Post Processing

In this chapter we describe the last step of the algorithm which, starting from the offset field obtained through the matching step, leads to the creation of the 3D detection map. For copy-move detection algorithms, this step is usually called Post-Processing Dense Linear Fitting (DLF) and false alarm removal are the main parts of our post-processing, described in the next two sections.

4.1 Dense Linear Fitting

Let NNF be the nearest neighbor field at the output of the matching algorithm. It could be expected to be mostly chaotic in the pristine regions of the video and very smooth, with linear behavior, in the regions where a copy-move occur. Actually, real NNFs do not follow so closely this model, causing missing detections in tampered regions and false alarms in pristine ones. The technique to avoid, or at least reduce, false alarms will be described in next section. Instead, to prevent missing detections we rely on a suitable regularization of the NNF. Many papers have treated the regularization problem and many sophisticated methods have been proposed in the context of copy-move detection (e.g. RANSAC [89], SATS [19]). These methods, however, are too slow for our needs. Furthermore, thanks to the behavior of our matching algorithm, the NNF is already smooth enough to perform a simpler regularization based on median filtering followed by dense linear fitting. The idea is to compute the dense linear fitting error of the NNF and use it to compute the detection map.

Let $\delta(s)$ be the offset field in a N-pixel neighborhood of the position s. We want to fit $\delta(s)$ with an affine model

$$\widehat{\delta}(s_i) = As_i, \;\; i = 1, \ldots, N \tag{4.1}$$

where A (the parameters of the transformation) is chosen to minimize the sum of squared errors w.r.t. $\delta(s)$

$$\epsilon^2(s) = \sum_{i=1}^{N} \|\delta(s_i) - \widehat{\delta}(s_i)\|^2 \tag{4.2}$$

Our offset field is 3-dimensional and the optimization problem can be solved one component at a time. So let us redefine $\delta(s)$ as a single component field. Now we can formulate the problem as

$$a^{\text{opt}} = \arg\min_a \|\delta - Sa\|^2 \tag{4.3}$$

In eq. 4.3 $\delta = [\delta(s_1), \delta(s_2), \ldots, \delta(s_N)]^T$ is the vector of offsets, $a = [a_0, a_1, a_2]^T$ is the vector of the transform's parameters and S is the $N \times 3$ matrix of the homogeneous coordinates of all pixels in the neighborhood

$$S = \begin{bmatrix} 1 & s_{11} & s_{12} \\ 1 & s_{21} & s_{22} \\ & \vdots & \\ 1 & s_{N1} & s_{N2} \end{bmatrix} \tag{4.4}$$

With this notation the affine model becomes

$$\widehat{\delta}(s_i) = a_0 + a_1 s_{i1} + a_2 s_{i2}, \;\; i = 1, \ldots, N \tag{4.5}$$

This is a multiple linear regression problem [59] with solution

$$a^{\text{opt}} = (S^T S)^{-1} S^T \delta \tag{4.6}$$

With this solution, the corresponding sum of squared errors (SSE) is associated

$$\begin{aligned} \epsilon^2(s) &= \|\delta - S(S^T S)^{-1} S^T \delta\|^2 \\ &= \|(I - H)\delta\|^2 \\ &= \delta^T (I - H)\delta \end{aligned} \tag{4.7}$$

where $H = S(S^T S)^{-1} S^T$. By choosing always the same shape of the neighborhood and taking the coordinates in (4.4) relative to s, the matrix H does not depend on s, so we can compute the SSE by evaluating the quadratic form (4.7).

In order to reduce the processing cost, we can consider that the rank-3 matrix H can be decomposed as:

$$H = QQ^T, \qquad Q = [q_1, q_2, q_3] \tag{4.8}$$

where q_j is a column vector of length N. This leads to calculate the SSE as

$$\epsilon^2(s) = (\delta^T\delta) - (\delta^T q_1)^2 - (\delta^T q_2)^2 - (\delta^T q_3)^2 \tag{4.9}$$

computed through a few filtering operations and some products.

4.1.1 Morphological Operations

We can now outline the process that leads to the creation of the 3D detection map (Fig.4.1), that is a firs version of the final map, as we will see in the next paragraph.

Let us consider the (squared) dense linear fitting error ϵ^2 and the smoothed version of nearest neighbor field, NNF_f, computed by median filtering the original NNF. Again, we make reference to the offset field, δ, that is, the NNF_s relative to each position s of the video. To this end, if S is a matrix such that $S(s) = s$, we define the offset

$$\delta = NNF_f - S; \tag{4.10}$$

Now, remember that ϵ^2 is a 3D matrix, with the same size as the video, and that each value of the matrix is a 3-component vector comprising the errors computed in the three dimensions of the video (ϵ_r^2, ϵ_c^2, ϵ_t^2). Therefore, we compute a binary map of the error, M_{error}, by thresholding a weighted sum of these error components

$$M_{error} = \left(\epsilon_r^2 + \epsilon_c^2 + \eta \cdot \epsilon_t^2\right) \leq T_{error} \tag{4.11}$$

where $\eta > 1$ is a parameter used to emphasize the error over the time dimension and T_{error} is a suitable threshold. Another binary map, $M_{distance}$, is then defined to discard matching between regions that are too close. To this aim, if D_{min} is the minimum acceptable distance for a copy move, we can write

$$M_{distance} = \|\delta\|_2 \geq D_{min} \tag{4.12}$$

A first version of the detection map, $\hat{M}$, is then obtained as the product[1] of these two maps

$$\hat{M} = M_{error} \otimes M_{distance} \tag{4.13}$$

[1]The product $\otimes$ in eq. 4.13 denotes the element-by-element multiplication.

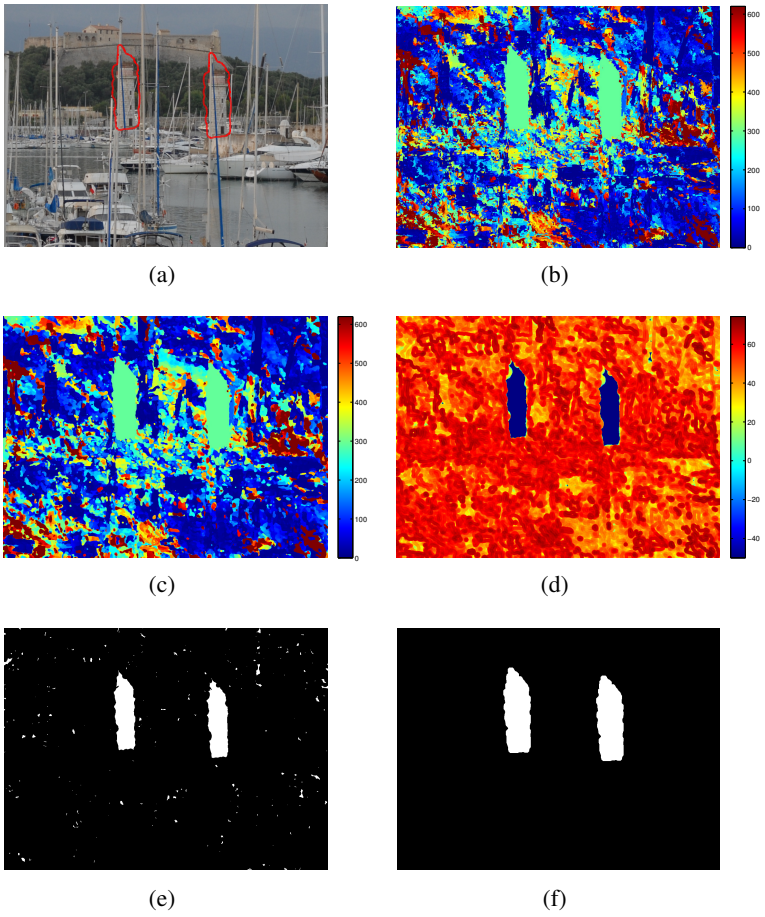


Figure 4.1: Post-processing steps: (a) forged image, (b) magnitude of offsets, (c) median filtering, (d) fitting error $\epsilon^2(s)$ (dB), (e) thresholding of $\epsilon^2(s)$, (f) final mask.

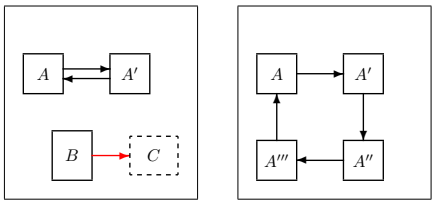


Figure 4.2: Left: removing random false matches. The preliminary detection map M^{DLF} includes two clones, A, A', and a false match B pointing to region C that has zero vlues in the detection map. Since B does not point to a detected region in the map it is eventually removed. Right: multiple clones. The four clones, A, A', A'', A''' all belong to the preliminary map. Even though no two regions match one another, they all match regions in the map, so they are all kept.

Unfortunately, spurious matchings abound in video sequences because of repeated patterns, uniform background and mainly stationary scenes. For these reasons $\hat{M}$ is typically characterized by several highlighted regions, which represent false alarms. However, such regions that do not correspond to a real copy-move are usually quite small, hence we can remove most of them through a simple morphological opening, with a circular structuring element of suitable radius. The resulting map, M^{DLF}, is the starting point for the last step of our post-processing, described in the next section.

4.2 Removing False Alarms

Pristine regions that are similar to one another abound in images and videos, and may induce a copy-move detection algorithm to produce false alarms. In [26], dealing with images, these were largely eliminated through morphological filtering. When dealing with videos, however, this problem becomes much more relevant because subjects, and especially background areas, appear almost identical in many subsequent frames, giving rise to a large number of false alarms, over extended regions. Standard morphological filtering cannot solve the problem anymore. We therefore add a further control, working always on the NNF to keep high efficiency.

The inspiring principle, already used for other aims [43, 37, 90], is that true clones should match both ways, that is

$$(s,t) + \delta(s,t) = (s',t') \;\Leftrightarrow\; (s',t') + \delta(s',t') = (s,t) \tag{4.14}$$

Points for which this condition does not hold may be random matches rather than corresponding points of a copy-move. Actually, this is too strict a condition to enforce, since small deviations from this rule apply also to actual copy-moves. In addition, with multiple clones, the very same principle weakens, as points may exhibit a circular matching. Therefore we use a weaker but still effective constraint, requiring simply that regions matching through the NNF all belong to preliminary detection map M^{DLF}. This is rarely the case for false matches, while it happens with near certainty for actual copy-moves. Pictorial examples of the application of these rules are shown in Fig.4.2 with reference to a 2D geometry.

In Fig.4.3, instead, we show some example detection maps for videos with copy moves before and after the application of the proposed technique for false alarm removal. In both cases the copy-moves are detected with good accuracy, but the original map also present a large number of false alarms, which are mostly removed with the specific post-processing.

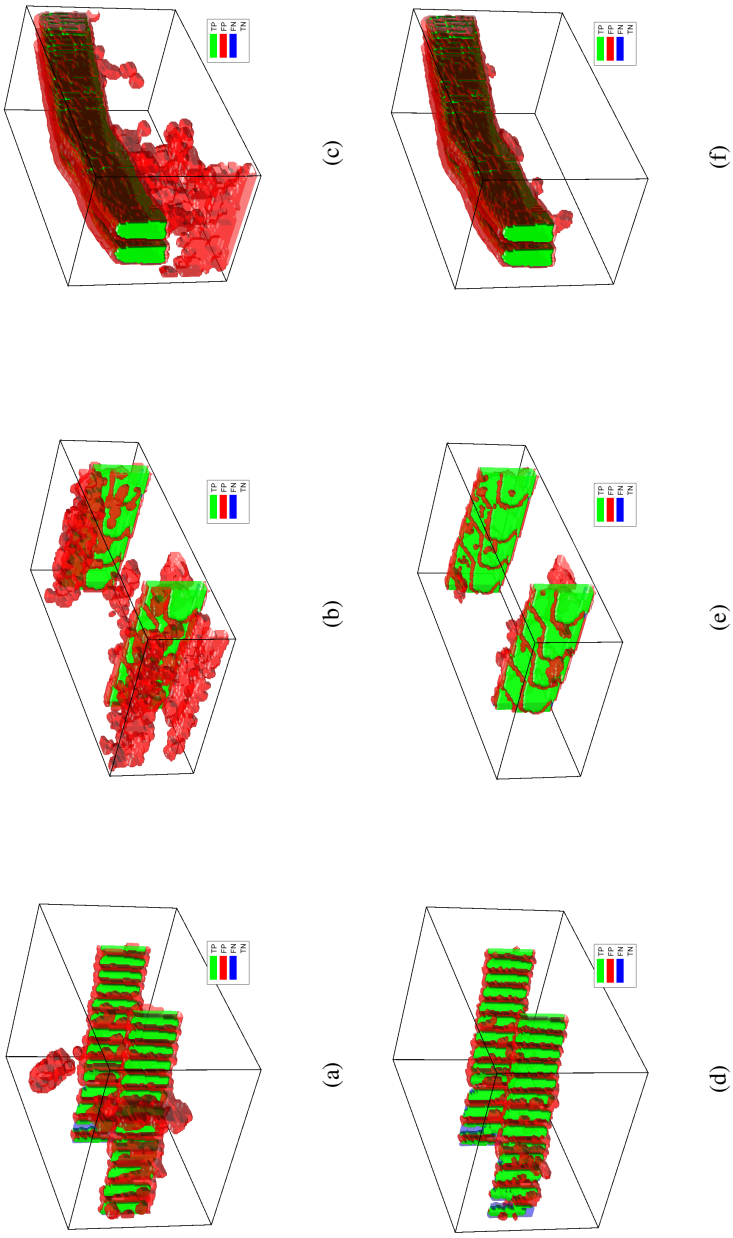


Figure 4.3: Examples of color-coded detection maps for videos with copy-moves, original (top) and after applying the technique for false alarm removal (bottom). TP pixels are green, FP pixels are red and FN pixels are blue. TN pixels are transparent for correct visualization.

Chapter 5

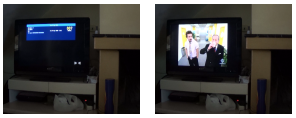
Experimental Results

The performance of the proposed method was assessed through a number of experiments under various operative conditions. In the following we will describe the datasets, the performance measures, and finally the experimental results.

5.1 The GRIP Dataset

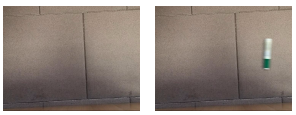
We prepared a dataset, called the GRIP dataset from now on, comprising 15 short videos with rigid copy-moves, 10 additive and 5 occlusive. They were obtained using After Effects Pro, a tool for video editing. As a result, there are little or no artifacts which may raise suspects on the video, just as would happen with a real-world skilled attacker. Nonetheless, since we consider rather short videos, additive copy-moves may be obvious anyway, since the same object appears twice within a few seconds. On the contrary, it seems safe to say that occlusive copy-moves can be hardly spotted without specific tools. In addition, by using rotation or temporal flipping, when meaningful, also additive copy-moves become less visible. All copy-moved videos are available online at http://www.grip.unina.it/ together with their pristine versions and the ground truths.

Tab.5.1 shows synthetic statistics of all videos and forgeries, while Figures 5.1, 5.2 and 5.3 show a sample frame of the original and forged video, together with a 3D view of the ground truth which provides some immediate insight on the spatio-temporal structure of the forgery. Large and static copy-moves, like that of TV Screen (video $\#1$) will be easily detected in any condition. On the contrary, small and fast-moving copy-moves represent a severe chal-

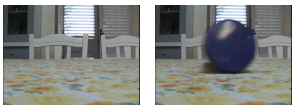


a) video #1, TV screen: additive, large, static

b) video #2, Fast car: additive, large, fast (low depth)



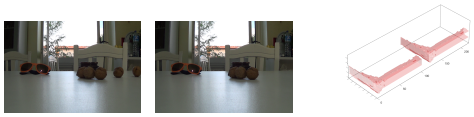
c) video #3, Felt-Tip Pen: additive, small



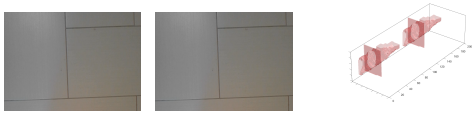
d) video #4, Rolling Can: additive, large

e) video #5, Falling can: additive, small, fast (low depth)

Figure 5.1: GRIP dataset: videos from #1 to #5. From left to right: original frame, copy-moved frame, 3D view of the ground truth. In the 3D views, the origin of the axes is the bottommost point, and the time axis is on bottom-right.

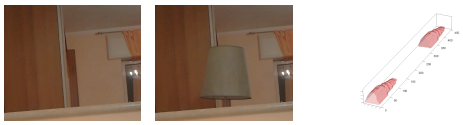


f) video #6, Walnuts: occlusive, small, saturated area

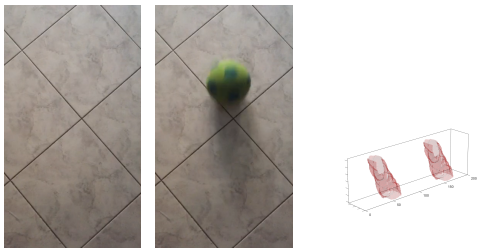


f) video #7, Can 1: additive, large

f) video #8, Can 2: additive, large



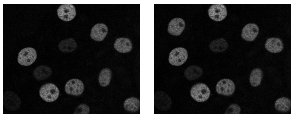
a) video #9, Lamp: additive, large



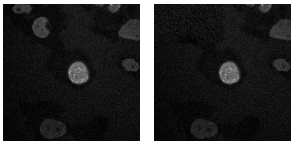
b) video #10, Ball: additive, large

Figure 5.2: GRIP dataset: videos from #6 to #10. From left to right: original frame, copy-moved frame, 3D view of the ground truth. In the 3D views, the origin of the axes is the bottommost point, and the time axis is on bottom-right.

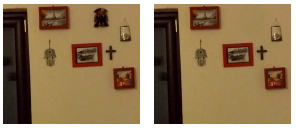
c) video #11, Student: occlusive



d) video #12, Cell 1: additive



e) video #13, Cell 2: occlusive



f) video #14, Wall frame: occlusive, small

f) video #15, Statue: occlusive

Figure 5.3: GRIP dataset: videos from #11 to #15. From left to right: original frame, copy-moved frame, 3D view of the ground truth. In the 3D views, the origin of the axes is the bottommost point, and the time axis is on bottom-right.

video				copy-move				
#	name	frame size	frames	add./occ.	$\rho_{\max}$	$d_{\max}$	rot.	flp.
1	TV screen	576×720	141	add	182.7	43	✓	✓
2	Fast Car	370×720	140	add	203.2	9		✓
3	Felt-Tip Pen	550×720	100	add	62.3	4	✓	✓
4	Rolling Can	480×660	125	add	229.6	18	✓	✓
5	Falling Can	480×720	174	add	71.3	29		
6	Walnuts	480×720	221	occlusive	199.5	102		
7	Can 1	520×720	201	add	220.6	28		✓
8	Can 2	720×720	210	add	112.1	15	✓	✓
9	Lamp	390×465	455	add	159.9	129	✓	
10	Ball	640×360	200	add	195.4	31		✓
11	Student	400×380	340	occlusive	176.3	60		
12	Cell 1	400×500	92	add	63.7	92	✓	✓
13	Cell 2	512×512	92	occlusive	107.5	92	✓	✓
14	Wall Frame	500×570	200	occlusive	50.6	155	✓	
15	Statue	590×480	100	occlusive	65.9	61		

Table 5.1: Features of the GRIP dataset

lenge. Note that by "fast", we mean a copy-move with a rapidly changing mask, maybe spanning just a few frames at any pixel, like in both Fast Car (video $\#2$) and Falling Can (video $\#5$), while the speed of the physical object inside the mask is immaterial for our aim. To capture synthetically these geometric features we use the max-radius and max-depth indicators. The max-radius is defined as $\rho_{\max} = \max_t \sqrt{A(t)/\pi}$, with $A(t)$ being the area of the tampered region in frame t. Likewise, max-depth is $d_{\max} = \max_s d(s)$, with $d(s)$ the depth of the tampered region for pixel s, possibly much smaller than the total copy-move duration. Walnuts (video $\#6$) and Student (video $\#11$) are examples of occlusive forgeries. The first one may be especially challenging, giving rise to a large number of false alarms due to the saturated area in the middle.

Besides our own dataset, we also consider the REWIND dataset, described

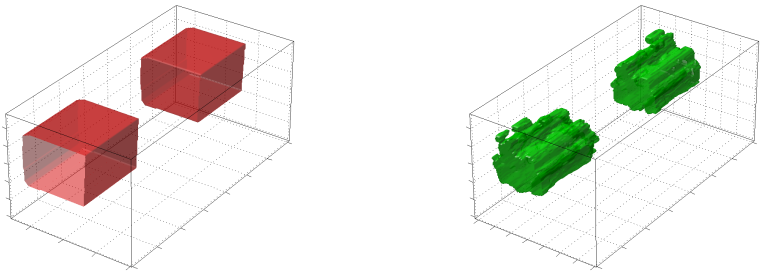


Figure 5.4: Examples of ground truth (GT) and detection map (M). In GT (left) copy-moved regions, both original and clones, are set to 1 (red). In M (right), detected copy-moves are set to 1 (green).

in [7] and available online[1]. This dataset, however, comprises only rigid additive copy-moves and comes without a ground truth. In addition, some videos (e.g., Duck, Fast Car) appear to be splicings rather then copy moves (maybe with material taken from parts of the video not available to the user) or else the copied regions have been subjected to some unreported processing before pasting them back. For these reasons, we use REWIND only for some experiments, turning to the GRIP dataset for a more reliable analysis.

5.2 Performance Assessment

The performance is measured in terms of detection and localization accuracy, besides processing time. Detection is declared if, after the post-processing, which includes the removal of small regions, a large number of detected pixels are present in the output map M

$$|M| > T_{\text{detection}} \tag{5.1}$$

where $|x|$ counts the number of ones in x and the threshold $T_{\text{detection}}$ is set for each version of the algorithm in order to maximize the detection's performance. When a copy-move is actually present, this is a correct detection, otherwise it is a false alarm. Therefore, to quantify both missing detections

[1] https://sites.google.com/site/rewindpolimi/downloads/datasets/video-copy-move-forgeries-dataset

and false alarms, in the experiments we run our detectors twice, first on the copy-moved video and then on the pristine original video.

To measure the localization performance we define the sets

- TP (true positive): detected copy-move pixels;
- FP (false positive): detected pristine pixels;
- TN (true negative): undetected pristine pixels;
- FN (false negative): missed copy-move pixels.

(see also Fig.5.5) from which the F-measure indicator is derived as

$$F = \frac{2|\mathrm{TP}|}{2|\mathrm{TP}| + |\mathrm{FP}| + |\mathrm{FN}|} \tag{5.2}$$

If detection map and ground truth coincide, then $|\mathrm{FP}| = |\mathrm{FN}| = 0$, and the F-measure reaches its maximum value, equal to 1. However, as the number of false negative or false positive pixels increases, the F-measure decreases rapidly. In particular, the F-measure is more informative than the overall accuracy when the two classes of interest are very unbalanced, which is the case of typical forged videos, where only a small fraction of the data are tampered with.

Finally, we measure efficiency in terms of normalized CPU-time, s/Mpixel, that is, the time in seconds required to process the whole video divided by its size in Mpixel. CPU-times refer to a computer with a 2GHz Intel Xeon processor with 16 cores, 64GB RAM and GPU Nvidia GeForce GTX Titan X.

	Basic 2D				Basic 3D				Fast 2D				Fast 3D				Bestagini			
video	det.	f.a.	F	time	det.	f.a.	F	time	det.	f.a.	F	time	det.	f.a.	F	time	det.	f.a.	F	time
1	✓		0.96	15.42	✓		0.95	17.50	✓		0.97	2.17	✓		0.97	3.25	✓		–	8.9
2	✓		0.88	15.45	✓		0.68	17.19	✓		0.78	2.77	✓		0.67	3.44	✓		–	7.3
3	✓		0.56	16.39	✓		0.29	23.24	✓		0.60	2.67	✓		0.31	3.00	✓		–	6.7
4	✓		0.88	14.92	✓		0.79	16.75	✓		0.88	2.77	✓		0.76	3.32	✓		–	7.2
5	✓		0.84	16.70	✓		0.86	20.29	✓		0.81	2.07	✓		0.86	3.24	✓	✓	–	14.9
6	✓	✓	0.72	16.50	✓		0.74	18.58	✓	✓	0.73	2.35	✓		0.81	3.45			–	11.7
7	✓		0.83	18.45	✓		0.78	20.25	✓		0.90	2.54	✓		0.81	3.41	✓	✓	–	11.5
8	✓		0.87	19.73	✓		0.77	24.23	✓		0.89	2.20	✓		0.76	3.32	✓	✓	–	15.2
9	✓		0.93	17.80	✓		0.92	20.31	✓		0.94	2.40	✓		0.93	4.02	✓	✓	–	14.4
10	✓		0.91	15.69	✓		0.89	16.67	✓		0.94	2.30	✓		0.92	3.45	✓	✓	–	6.3
11	✓	✓	0.88	14.14	✓		0.87	18.00	✓	✓	0.86	3.05	✓		0.88	4.15			–	7.7
12	✓		0.80	16.23	✓		0.77	18.78	✓		0.87	1.96	✓		0.83	3.81			–	2.6
13	✓		0.91	15.43	✓		0.90	18.26	✓		0.92	2.49	✓		0.91	4.02			–	4.4
14	✓		0.74	16.66	✓		0.71	19.42	✓		0.77	2.35	✓		0.77	3.39			–	8.8
15	✓		0.72	16.05	✓	✓	0.51	20.17		✓	0.00	2.26	✓	✓	0.41	3.32			–	3.8
Σ, μ	15	2	0.83	16.37	15	1	0.76	19.31	14	3	0.79	2.42	15	1	0.75	3.51	9	5		8.8

Table 5.2: Detection, localization and efficiency performance for plain copy-moves on the GRIP dataset

			Basic 2D			Basic 3D			Fast 2D			Fast 3D			Bestagini		
dataset	case	# videos	det.	f.a.	F	det.	f.a.	F	det.	f.a.	F	det.	f.a.	F	det.	f.a.	F
GRIP	plain	15	15	2	0.83	15	1	0.76	14	3	0.79	15	1	0.75	9	5	–
GRIP	QP = 10	15	15	1	0.84	15	1	0.77	14	2	0.74	14	1	0.75	9	5	–
	QP = 15		15	1	0.76	15	1	0.72	13	2	0.65	15	1	0.70	9	4	–
	QP = 20		12	1	0.54	12	1	0.56	13	2	0.53	12	0	0.52	9	5	–
GRIP	$\theta = 5^o$	8	8	–	0.81	7	–	0.73	5	–	0.40	7	–	0.68	2	–	–
	$\theta = 25^o$		7	–	0.71	4	–	0.60	3	–	0.25	4	–	0.44	2	–	–
	$\theta = 45^o$		5	–	0.56	4	–	0.43	2	–	0.12	4	–	0.43	2	–	–
GRIP	flipping	9	8	–	0.81	9	–	0.76	6	–	0.59	7	–	0.59	3	–	–
REWIND	plain	10	8	4	–	9	4	–	8	4	–	6	1	–	6	3	–

Table 5.3: Summary of detection and localization performance on the whole set of experiments

5.2.1 Numerical results

In Tab.5.2 we report results for the GRIP dataset in the presence of plain copy-moves, involving only rigid spatio-temporal translations, and possibly some local processing at the boundary of the copied area to reduce artifacts. No rotation or flipping are allowed, here, and no global post-processing, such as compression, and noise addition. For each technique and each video we mark with a ✓ symbol whether the copy move is detected (det), and whether a false alarm (f.a.) is declared, namely a copy-move is detected in the pristine video where there are none. Then we report the F-measure, to quantify localization accuracy, and the normalized CPU-time. We use features of the length 12, in the 2D case, and length 18, in the 3D case, the latter obtained by considering 6 Zernike moments over 3 consecutive frames and computing the even-odd transform. In the fast versions, with the multi-resolution analysis scheme, a subsampling step $S = 4$ is used. To provide some comparison with the state-of-the-art, we include also the technique proposed by Bestagini *et al.* [7] which, however, addresses only the detection task.

Unfortunately, other literature techniques like [50] and [100] make very restrictive hypotheses on the forged videos, hence they cannot be used on realistic datasets as GRIP or REWIND. In fact, in [50] only rigid copy-moves between consecutive frames are considered, while in [100] matching is carried out through an exhaustive search on dense HOG features, making the procedure unfeasible even for very short videos.

Performance figures are very good for all variants of the proposed algorithm. The basic version of the algorithm, without multi-resolution processing, detects all copy-moves, both with single-frame 2D features (Basic-2D algorithm, from now on) and with 3D flip-invariant features (Basic-3D), with very few false alarms. On the other hand, a few false alarms in this context are not really critical since they raise attention on some candidate copy-moves that may be analyzed with much greater care afterwards. On the contrary, missed detection cannot be recovered easily. The fast versions of the algorithms (Fast-2D and Fast-3D) are also quite reliable. Only Fast-2D misses one copy-move, of the occlusive type, probably because of the loss of spatial synchronization due to subsampling. The reference method [7], instead, misses all occlusive copy-moves and also an additive one, besides originating a slightly larger number of false alarms.

The localization performance is extremely high in all cases. Barring video 15, missed by Fast-2D, the only critical case seems to be video 3, and only for the Basic-3D and Fast-3D algorithms. This is easily explained by noting that

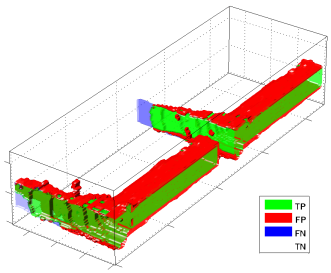
$d_{\max} = 4$, for this video, namely, the copy-move is extremely thin in time, causing inaccuracies at the temporal boundaries when 3D features are used. Nonetheless, these problems do not prevent correct detection.

Turning to processing speed, the overall CPU times scale proportionally with the video size (frame size $\times$ number of frames). In fact, the Basic 2D and 3D algorithms take about 16.4 s/Mpixel and 19.3 s/Mpixel, respectively, with very small deviations across the videos. The fast versions are indeed much faster, bringing the average CPU-time down to 2.9 and 3.5 s/Mpixel, respectively, the difference mainly due to the longer 3D features.

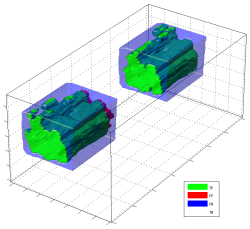
Let us now analyze performance in more challenging situations, namely, in the presence of video compression, and of copy-move rotation and flipping. Experimental results are reported in Tab.5.3, only in synthetic form for the sake of brevity. In the same table we also report results obtained in the absence of further processing on the GRIP dataset (GRIP plain, first line, reported again for completeness) and on the REWIND dataset (REWIND plain, last line). Instead, we do not show CPU-times anymore, since they depend almost exclusively on the video size, and very little on the level of compression or the type of attack.

Compressed videos are more the norm than the exception, and studying performance in this situation is of paramount importance. To this end, we consider MPEG-2 compression at quantization parameter QP = 10, 15, and 20, which correspond roughly to high, medium and low quality sources. Together with the forged videos, we compress also the pristine ones, which allows us to compute both detection and false alarm figures. The basic version of the algorithm keeps providing an excellent performance with both 2D and 3D flip-invariant features, although some missed detections are observed in the most challenging case of QP = 20. A similar behavior is observed with the fast versions, with just a few further missed detections. The F-measure remains always quite large (lower values are mostly due to missed detections), indicating a very good localization ability. In all cases, a very small number of false alarms is observed. Note that the reference technique detects only 9 copy-moved videos, with a larger number of false alarms and, as already said, does not have localization ability.

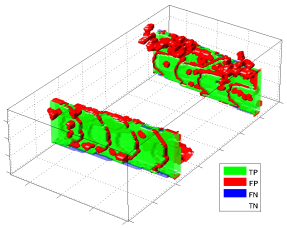
In lines 5-7 of Tab.5.3 we analyze the case of video objects that are rotated before pasting, which may be due to composition needs (small angles) or made on purpose to fool copy-move detectors. The analysis applies only to the 8 videos with rotated copy-moves, while no false alarm analysis is possible, since objects are not rotated in the original videos. The basic algorithm keeps



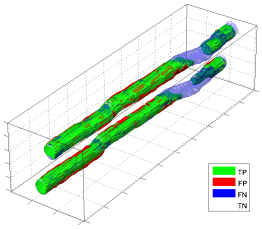
(a) Plain copy-move in video #6 Walnuts



(b) Copy-move with compression at QP=20 in video #1 TV screen



(c) Copy-move with flipping in video #2 Fast Car



(d) Copy-move with 45^o rotation in video #14 Wall Frame

Figure 5.5: Sample color-coded detection maps for videos of the GRIP dataset. TP pixels are green, FP pixels are red and FN pixels are blue. TN pixels are transparent for correct visualization. The plain (top-left) and flipped (bottom-left) copy moves are fully detected, with only some inaccuracies at the boundaries. Compression (top-right) impacts on localization performance, especially on the saturated areas of the video object. Rotation (bottom-right) causes the loss of a copy moved segment, following sudden camera motion.

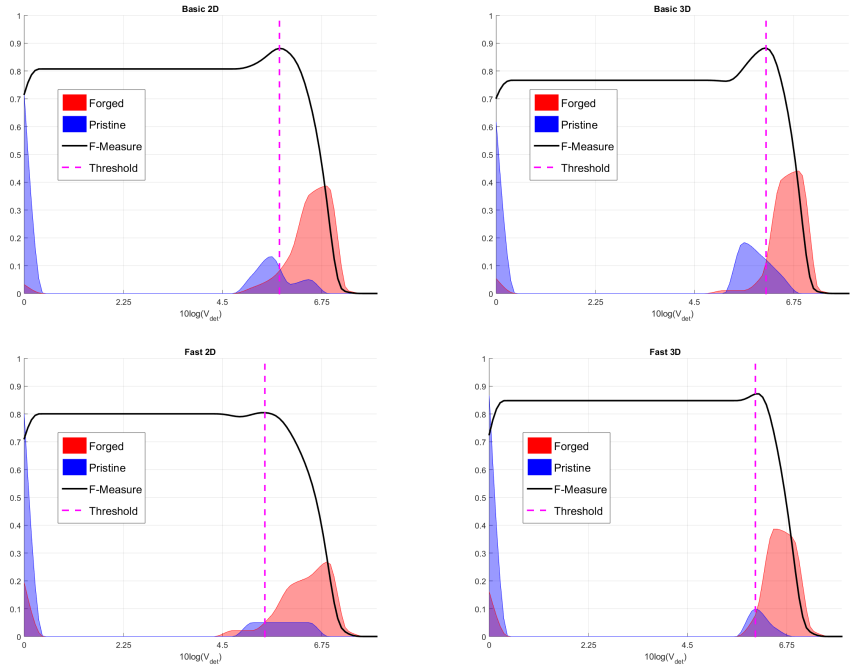


Figure 5.6: Analisys of performance over the whole test set as a function of the detection threshold. Each graph shows the distribution of the decision variable for forged videos (red) and pristine videos (blue), the corresponding F-measure (black line), and the selected threshold (magenta dashed line). Basic-2D (top-left), Basic-3D (top-right), Fast-2D (bottom-left), Fast-3D (bottom-right).

working very well at small angles, while at large angles, 25 or 45 degrees, the version with 3D features exhibits a large number of missed detections. This is likely due to temporal boundary discontinuities, quite relevant for videos with small $d_{\max}$. On the other hand, 3D features seem more robust when moving to the fast version, with no further missed detection, while the Fast-2D algorithm exhibits a limited reliability. The reference algorithm, instead, looks definitely unreliable with rotated copy-moves at all angles. In the presence of flipping, the proposed algorithm works very well with 3D flip-invariant features, with no missed detection for the basic version and only two of them for the fast version, slightly better than when 2D features are used. Together with previous results, this suggests using 3D flip-invariant features with the fast algorithm,

while 2D features seem slightly preferable with the basic algorithm. Finally, let us consider the REWIND dataset. In this case, no algorithm is able to provide perfect detection. The best result is obtained with Basic-3D, but even in this case there is one missed detection and four false alarms. However, this is to be ascribed to the video themselves since some forgeries appear to be splicings rather than copy-moves (notably the fast car video), which fully justifies the failures. Indeed, the reference algorithm, tested by the authors on this very same dataset, provides an even poorer performance. In Fig.5.5 we show some sample detection maps obtained with the proposed algorithm (basic, 2D features) on GRIP videos with copy-moves and various operating conditions. The performance is always very good, although a whole segment is missed in the rotated copy-move, due to the large rotation angle, highlighting the challenges raised by post-processing for detection.

All above results have been obtained by selecting the detection threshold which maximizes the F-measure computed on the whole dataset. As shown in Fig.5.6, the four versions of the algorithm are quite robust w.r.t. errors in the selection of the threshold. In particular, a threshold much smaller than the optimal can be used with an F-measure loss of approximately 0.1 for the Basic versions, and largely negligible for the Fast versions.

In Fig.5.7 we show the receiver operating curves (ROCs) obtained on the whole dataset by varying the detection threshold of eq.(5.1) for the proposed technique compared with the technique proposed in [7]. Besides the good performance, the curves testify to a good robustness to small errors in the selected threshold. Fig.5.8, instead, compares the ROCs obtained with and without the new post-processing described in Section 4.2 to reduce false alarms. The performance improvement is limited but consistent. Figures 5.9 through 5.10 illustrate the behavior of the proposed algorithm in various situations of interest by displaying the output 3D detection maps for some videos of the GRIP dataset. In particular, Fig.5.9 shows results for plain copy-moves and illustrates the effect of false positives on performance. Fig.5.10 compares the output maps obtained for the same video with and without copy-move. Fig.5.11 shows results for copy-moves with rotation, considering several rotation angles and focusing on the worst case for the algorithm. Finally, Fig.5.12, illustrates the technique's behavior in the presence of various levels of compression.

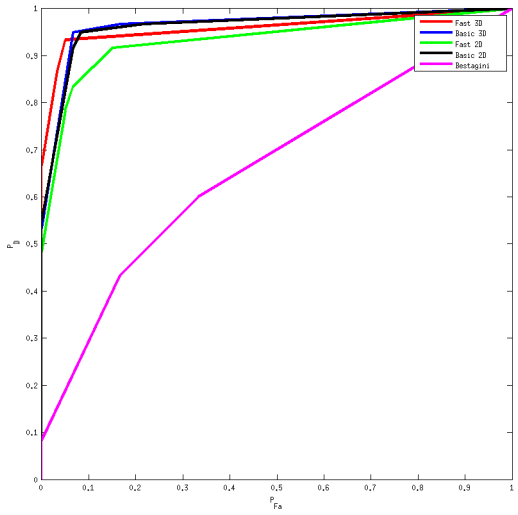


Figure 5.7: ROCs comparison between proposed technique and [7]: the sum of ones of the detection map is used as detection variable to generate these ROCs. To carry out a meaningful comparison, we considered only videos with forgeries that are detectable with the method proposed in [7], i.e. plain copy-moves and copy-moves with compression.

5.3 Complexity

In the development of the proposed algorithm, a major effort has been devoted to computational efficiency. The bar graphs of Fig.5.13 describe the results for the case of 2D features (left) and 3D flip-invariant features (right) in terms of normalized CPU times (s/Mpixel) averaged over all experiments, comprising a grandtotal of 153 videos. From left to right, the bars refer to the basic algorithm, its multi-resolution version, and the parallel implementation of the latter, while colors identify the phases of feature extraction (blue), matching (green) and post-processing (red). The multi-resolution processing impacts only on the matching phase, largely reducing its cost and bringing total CPU-time from 16.85 to 5.99 s/Mpixel with 2D features and from 20.06 to 7.42 s/Mpixel with 3D features. The parallel implementation, instead, reduces the cost of all phases, although to different degrees, bringing the total CPU-time to 2.43 and 3.47 s/Mpixel, respectively. Overall, Fast-2D guarantees a 7×

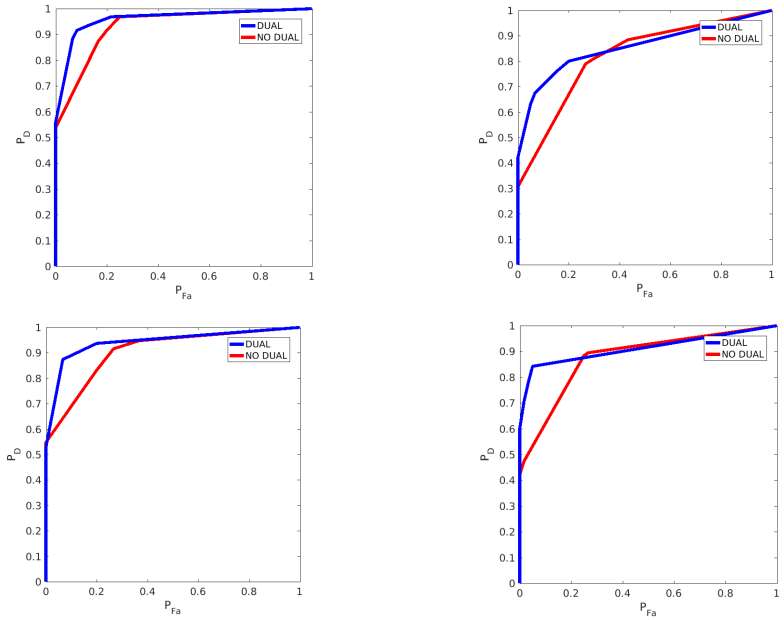


Figure 5.8: ROCs for the Basic-2D (top-left), Fast-2D (top-right), Basic-3D (down-left) and Fast-3D (down-right) algorithms, obtained by varying the detection threshold. In both cases, the post-processing used in this work (DUAL, blue line) provides significant improvements with respect to the simpler post-processing (NO-DUAL, red line).

speed-up w.r.t. Basic-2D, and Fast-3D a 6× speed-up w.r.t. Basic-3D, with differences due mainly to the longer features used in the second case.

It should be realized that this is a huge time saving with respect to plain search. Indeed, the complexity of copy-move detection is inherently quadratic with the length of the video, since, in principle, all features must be compared with one another. For the small videos of the GRIP dataset, one should compute in the order of 10^7 distances *per feature*. Thanks to PatchMatch, our basic algorithms reduce this number to about 10^2. Then, our fast parallel version is 6-7 times faster than that. Noteworthy, the method proposed in [7], based on Fourier-domain analysis, is much slower than Fast-2D and Fast-3D, and the same applies to a simple 3D version of the keypoint-based method proposed in [1].

All this said, the proposed algorithms are still computation-intensive and far from achieving real-time processing. To process a 1-minute video at 25

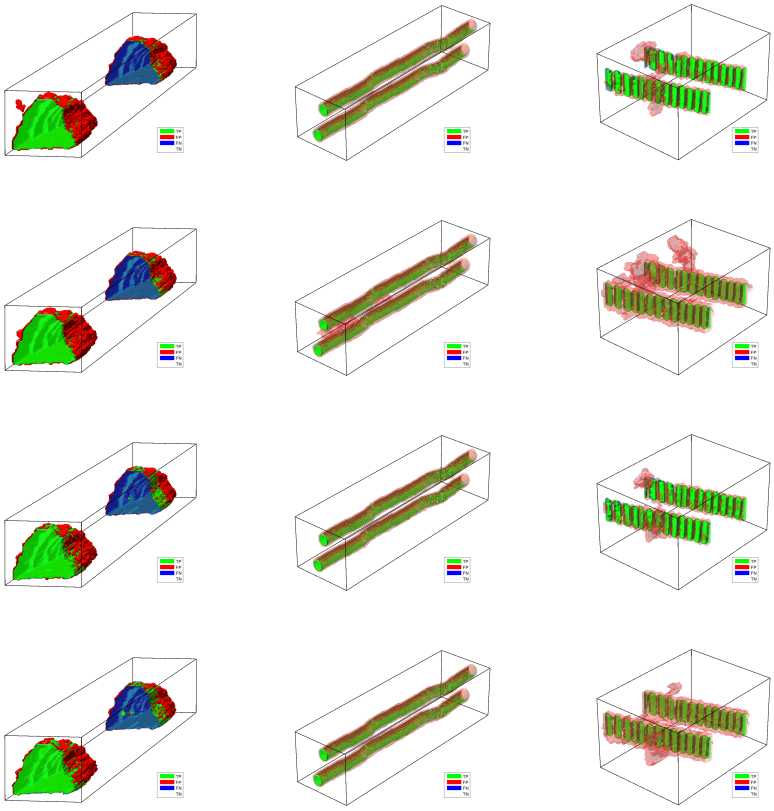


Figure 5.9: Examples of color-coded detection maps for plain copy-moves. TP pixels are green, FP pixels are red and FN pixels are blue. TN pixels are transparent for correct visualization. From left to right: videos #9, #14 and #3. From top to bottom: results of Basic-2D, Basic-3D, Fast-2D, Fast-3D techniques. For video #3, the presence of false positives (red in the maps) causes a reduction of the F-measure, as shown in Tab.5.2.

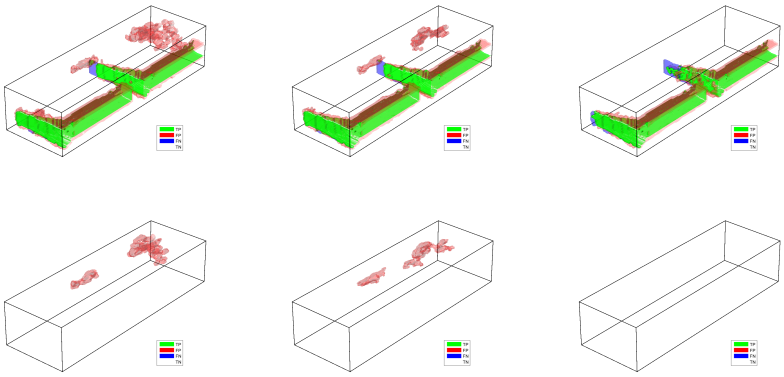


Figure 5.10: Examples of color-coded detection maps for forged vs. pristine video #6. TP pixels are green, FP pixels are red and FN pixels are blue. TN pixels are transparent for correct visualization. From left to right: results of Basic-2D, Fast-2D, Fast-3D techniques. Top: maps for the forged video; bottom: maps for the pristine video. False positives on the pristine video produce false alarms, as shown in Tab.5.2.

frames/s, with 0.5 Mpixel frames, about 30 minutes of CPU time are currently necessary. Obviously, much faster methods can be conceived, trading off speed for reliability. To explore this opportunity, we tested a bare-bone version of the proposed algorithm where *i)* the video is subsampled with step 16 in both spatial directions; *ii)* simpler non-flip invariant 2D features are used; and *iii)* PatchMatch relies only on vertical and horizontal predictors for propagation. With these simplifications, a dramatic 40$\times$ speed-up is obtained with respect to Fast-3D, reaching 0.08 s/Mpixel as opposed to 3.47 s/Mpixel. However, the decline in performance is not acceptable. Restricting attention to plain copy-moves, we observe a good detection rate, nearly 90%, even in the presence of compression, but a false alarm rate that grows to 20%, in the absence of compression, and to 40%, with compression. Therefore, given 1000 pristine videos, this tool would select from 200 to 400 of them for further analysis, hardly a saving of resources for the human analysts. Moreover, for copy-moves with flipping the detection rate drops to about 50%, and nearly to zero for copy-moves with rotation, allowing a smart attacker to easily fool the detector through a very small rotation of the copy.

In any case, in many circumstances, reliability is far more important than

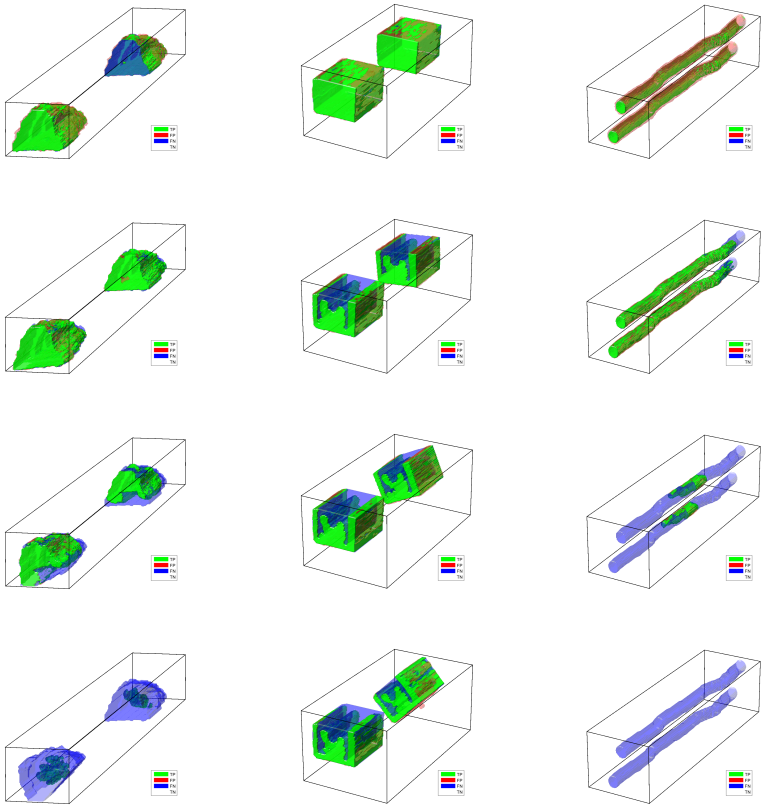


Figure 5.11: Examples of color-coded detection maps for copy-moves with rotation. TP pixels are green, FP pixels are red and FN pixels are blue. TN pixels are transparent for correct visualization. From left to right: videos #9, #1 and #14. From top to bottom: results for rotations of 5, 25 and 45 degrees. Results are obtained with the Fast-2D technique, the worst case for rotations, as shown in Tab.5.3.

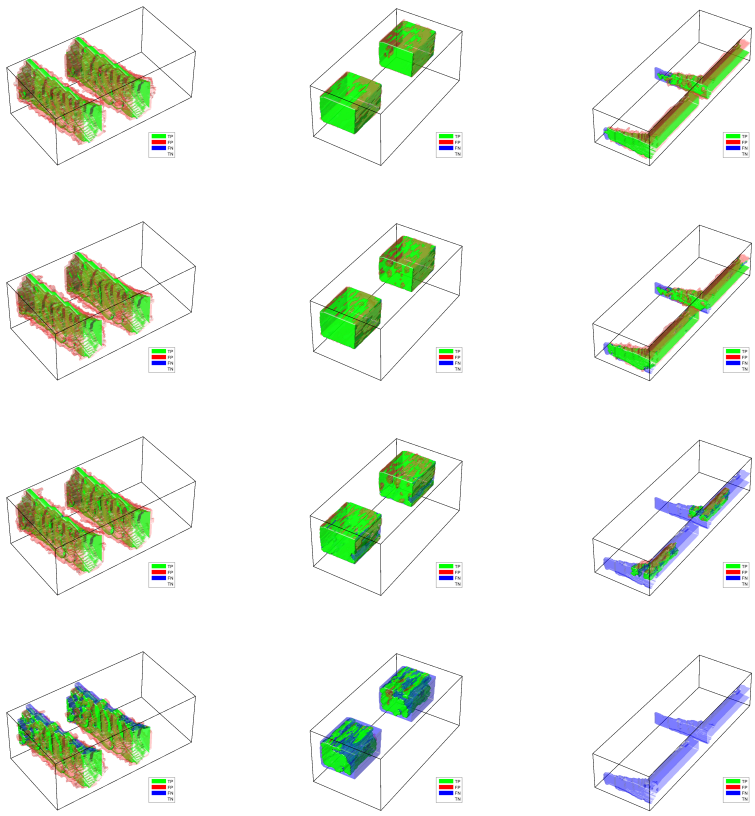


Figure 5.12: Examples of color-coded detection maps for copy-moves after compression. TP pixels are green, FP pixels are red and FN pixels are blue. TN pixels are transparent for correct visualization. From left to right: videos #4, #1 and #6. From top to bottom: no compression, compression with QP=10, QP=15 and QP=20. Results are obtained with the Fast-3D technique, but represent well all other versions of the technique. Video #6 represents the worst case in the presence of compression due to the type of forgery, that is an occlusive copy-move covering a relatively small region.

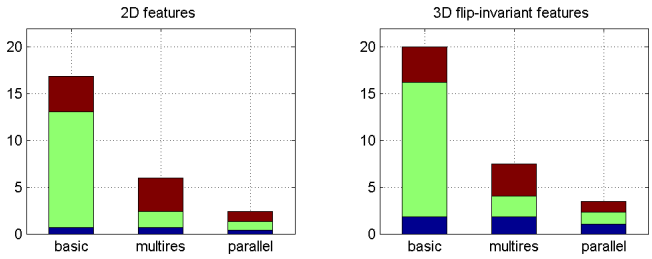


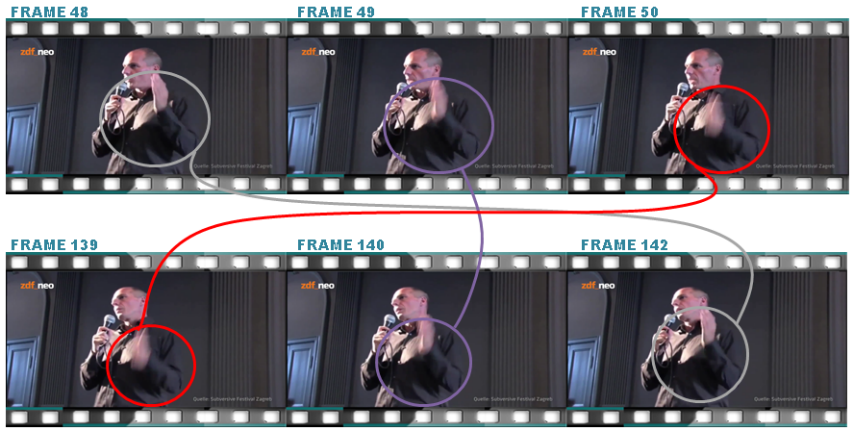
Figure 5.13: Computational cost of feature extraction (blue), matching (green), and post-processing (red) phases for various versions of the proposed algorithm using 2D features (left) or 3D flip-invariant features (right). The dominant source of complexity in the basic algorithm (bar 1) is matching. Multiresolution processing (bar 2) reduces sharply this cost. Parallel computing (bar 3) further reduces the cost of all phases. The final speed-up w.r.t. the basic version is about 7× with 2D features and 6× with 3D flip-invariant features.

speed, especially in forensic applications. In addition, often one has already selected a fragment of the video for verification, because of its semantics (see next subsection), and wants to analyze the rest of the video to locate regions that match the target fragment. In this modality, the search complexity reduces from quadratic to linear in the length of the video, with a major impact on efficiency.

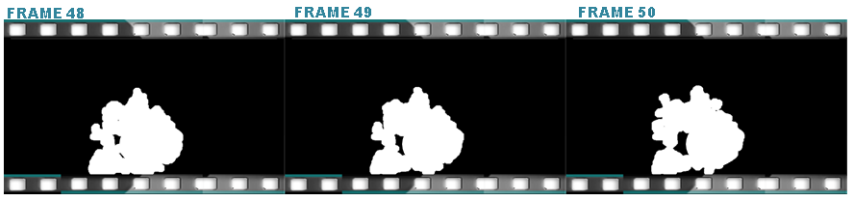
5.4 A real-world case: the Varoufakis video

We tested our algorithm on a real-world case that recently made the headlines all over the world, the well-known Varoufakis video. While politicians of the European Union (EU) were actively addressing the Greek financial crisis, in early 2015, a video was posted on Youtube™ with the greek minister of economy, Yanis Varoufakis, apparently "sticking the middle finger" at Germany to underscore his disappointment about the proposed EU economic recipes. The video become immediately a diplomatic case. Although minister Varoufakis quickly denounced the video as a fake, doubts persisted over its real nature, as it was impossible to discover clear signs of manipulations. The case became

Figure 5.14: Frames taken from the three Varoufakis videos. From top to down: #1, sticking middle finger, #2, arm down, #3, victory sign.



(a) Matching frames



(b) Detection maps

Figure 5.15: Findings in the Varoufakis video #2 (arm down). Top, evidence of copy-move with flipping. Bottom, sample detection maps.

even more complicated when two more versions of the same video appeared[2], one with the minister's arm down, and another one with raised arm but two fingers sticking in a victory sign. Frames extracted from the three videos are shown in Fig.5.14. Obviously, at least two of the videos had been manipulated. We therefore applied our algorithm to the videos in search of clues of what really happened. Although the videos were several minutes long, the sequence with the raised finger, where the videos differ, lasted just a few seconds, so we could adopt an asymmetric modality of analysis, focusing only to this section and looking for possible matching in the rest of the video. This circumstance made the computational effort fully acceptable. The proposed algorithm did not discover any copy-moves in videos #1 (middle-finger) and #3 (victory). Since only one of them (at most) can be pristine, we are missing a forgery. A

[2]see http://henryjenkins.org/2015/08/f-for-fake-in-the-second-order-yanis-varoufakis-the-germans-and-the-middle-finger-that-wasnt-there.html for a full account.

first possible explanation is that the victory video is original, and the other one is obtained by hiding the index finger through inpainting, very easy on small areas. However, it is also possible that the middle-finger video is original, and the victory sign is obtained by copy-moving the index from somewhere else. However, for such a small copy-move detection becomes very unlikely for any algorithm. The proposed algorithm was instead able to detect a clear forgery in video #2 (arm down), a copy-move with flipping from a temporally close section of the same video. Fig.5.15 shows the relevant frames, with the matching regions, and the corresponding detection maps. To obtain a visual confirmation of this finding, we played two instances of video #2 side by side, one going forward and the other backward in time. With suitable synchronization, the copy-move appeared obvious, and could easily pass the scrutiny of a court of justice. Therefore, the proposed method seems to work also outside the laboratory, barring prohibitive conditions where any algorithm would fail.

Conclusion

We have proposed a method for the detection and localization of video copy-moves. Since keypoint-based approaches are ineffective with most occlusive forgeries, we focused on dense-field methods. With this approach, the main issue is complexity, especially for videos, cursed by their huge data size. To deal with this problem we resorted to a fast randomized patch matching algorithm, a hierarchical analysis strategy, and parallel implementation. Experiments confirm that the proposed method has an excellent detection and localization ability, also for occlusive copy-moves, and even in adverse scenarios including rotated copy-moves and compressed videos. Moreover, the running time is much reduced w.r.t. linear search, enabling practical video analysis.

Despite all efforts, the proposed method cannot be used for real-time analysis or mass screening of video repositories. Therefore, there is much room for future research on tools that solve these problems, even at the price of reduced reliability. We ourselves are currently working on the development of fast keypoint-based methods for video analysis, and on the integration of PatchMatch with fast nearest neighbor search algorithms [107].

Bibliography

[1] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A sift-based forensic method for copy-move attack detection and transformation recovery," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1099–1110, Sep. 2011.

[2] A. Babenko and V. Lempitsky, "The inverted multi-index," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3069–3076.

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics*, vol. 28, no. 3, pp. 24:1–24:11, Jul. 2009.

[4] C. Barnes, E. Shechtman, D. Goldman, and A. Finkelstein, "The generalized patchmatch correspondence algorithm," in *European Conf. on Computer Vision*, vol. 6313, 2010, pp. 29–43.

[5] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[6] S. Bayram, H. Sencar, and N. Memon, "An efficient and robust method for detecting copy-move forgery," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Apr. 2009, pp. 1053–1056.

[7] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro, "Local tampering detection in video sequences," in *IEEE International Workshop on Multimedia Signal Processing*, October 2013, pp. 488–493.

[8] T. Bianchi and A. Piva, "Image forgery localization via block-grained analysis of jpeg artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1003–1017, Jun. 2012.

[9] A. Bidokhti and S. Ghaemmaghami, "Detection of regional copy/move forgery in mpeg videos using optical flow," in *Artificial intelligence and signal processing (AISP), 2015 International symposium on*. IEEE, 2015, pp. 13–17.

[10] G. K. Birajdar and V. H. Mankar, "Digital image forgery detection using passive techniques: A survey," *Digital Investigation*, vol. 10, no. 3, pp. 226–245, 2013.

[11] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, "Determining image origin and integrity using sensor noise," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74–90, March 2008.

[12] 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*, in press 2015.

[13] Y.-L. Chen and C.-T. Hsu, "Detecting recompression of jpeg images via periodicity analysis of compression artifacts for tampering detection," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp. 396–406, Jun. 2011.

[14] G. Chetty, "Blind and passive digital video tamper detection based on multimodal fusion," in *Proc. of the 14th WSEAS International Conference on Communications*, 2010, pp. 109–117.

[15] G. Chierchia, S. Parrilli, G. Poggi, L. Verdoliva, and C. Sansone, "Prnu-based detection of small-size image forgeries," in *Digital Signal Processing (DSP), 2011 17th International Conference on*. IEEE, 2011, pp. 1–6.

[16] G. Chierchia, G. Poggi, C. Sansone, and L. Verdoliva, "A Bayesian-MRF approach for PRNU-based image forgery detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 4, pp. 554–567, April 2014.

[17] G. Chierchia, D. Cozzolino, G. Poggi, C. Sansone, and L. Verdoliva, "Guided filtering for prnu-based localization of small-size image forgeries," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6231–6235.

[18] G. Chierchia, G. Poggi, C. Sansone, and L. Verdoliva, "Prnu-based forgery detection with regularity constraints and global optimization," in *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*. IEEE, 2013, pp. 236–241.

[19] V. Christlein, C. Riess, and E. Angelopoulou, "On rotation invariance in copy-move forgery detection," in *IEEE International Workshop on Information Forensics and Security*, December 2010.

[20] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, "An evaluation of popular copy-move forgery detection approaches," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1841–1854, Dec. 2012.

[21] V. Conotter, J. F. O'Brien, and H. Farid, "Exposing digital forgeries in ballistic motion," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 283–296, 2012.

[22] D. Cozzolino, G. Poggi, and L. Verdoliva, "Copy-Move forgery detection based on PatchMatch," in *IEEE International Conference on Image Processing (ICIP)*, Oct. 2014, pp. 5312–5316.

[23] D. Cozzolino, "Image forgery detection and localization," 2015.

[24] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery detection through residual-based local descriptors and block-matching," in *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5297–5301.

[25] D. Cozzolino, F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, "Prnu-based forgery localization in a blind scenario," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 569–579.

[26] D. Cozzolino, G. Poggi, and L. Verdoliva, "Efficient dense-field copy–move forgery detection," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 11, pp. 2284–2297, 2015.

[27] ——, "Splicebuster: A new blind image splicing detector," in *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*. IEEE, 2015, pp. 1–6.

[28] ——, "Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection," *arXiv preprint arXiv:1703.04615*, 2017.

[29] D. Cozzolino and L. Verdoliva, "Single-image splicing localization through autoencoder-based anomaly detection," in *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*. IEEE, 2016, pp. 1–6.

[30] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on image processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[31] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[32] L. D'Amiano, D. Cozzolino, G.Poggi, and L. Verdoliva, "Video forgery detection and localization based on 3D PatchMatch," in *IEEE International Conference on Multimedia and Expo Workshops*, 2015, pp. 1–6.

[33] L. D'Amiano, D. Cozzolino, G. Poggi, and L. Verdoliva, "A patchmatch-based dense-field algorithm for video copy-move detection and localization," *arXiv preprint arXiv:1703.04636*, 2017.

[34] S. Das, G. Darsan, L. Shreyas, and D. Devan, "Blind detection method for video inpainting forgery," *International Journal of Computer Applications*, vol. 60, no. 11, 2012.

[35] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004, pp. 253–262.

[36] D. D'Avino, D. Cozzolino, G. Poggi, and L. Verdoliva, "Autoencoder with recurrent neural networks for video forgery detection," in *IS&T International Symposium on Electronic Imaging: Media Watermarking, Security, and Forensics*, 2017.

[37] G. Egnal and R. P. Wildes, "Detecting binocular half-occlusions: Empirical comparisons of five approaches," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 8, pp. 1127–1133, 2002.

[38] H. Farid, "Image forgery detection," *IEE Signal Processing Magazine*, vol. 26, no. 2, pp. 16–25, Mar. 2009.

[39] C. Feng, Z. Xu, W. Zhang, and Y. Xu, "Automatic location of frame deletion point for digital video forensics," in *ACM workshop on Information hiding and multimedia security*, 2014, pp. 171–179.

[40] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva, "Image forgery localization via fine-grained analysis of cfa artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1566–1577, Oct. 2012.

[41] J. Fridrich, D. Soukal, and J. Lukás, "Detection of copy-move forgery in digital images," in *proc. of Digital Forensic Research Workshop*, 2003.

[42] H. Fu and X. Cao, "Forgery authentication in extreme wide-angle lens using distortion cue and fake saliency map," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1301–1314, Aug. 2012.

[43] P. Fua, "A parallel stereo algorithm that produces dense depth maps and preserves image features," *Machine vision and applications*, vol. 6, no. 1, pp. 35–49, 1993.

[44] A. Gironi, M. Fontani, T. Bianchi, A. Piva, and M. Barni, "A video forensic technique for detection frame deletion and insertion," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2014, pp. 6226–6230.

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

[46] M. Granados, K. Kim, J. Tompkin, J. Kautz, and C. Theobalt, "Background inpainting for videos with dynamic objects and a free-moving camera," in *European Conference on Computer Vision (ECCV)*, 2012, pp. 682–695.

[47] M. Granados, J. Tompkin, K. Kim, O. Grau, J. Kautz, and C. Theobalt, "How not to be seen object removal from videos of crowded scene," in *Computer Graphics Forum 31*, 2012, pp. 219–228.

[48] J. Hays and A. A. Efros, "Scene completion using millions of photographs," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 4.

[49] 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*, vol. 35, pp. 55–66, 2016.

[50] C.-C. Hsu, T.-Y. Hung, C.-W. Lin, and C.-T. Hsu, "Video forgery detection using correlation of noise residue," in *IEEE International Workshop on Multimedia Signal Processing*, 2008, pp. 170–174.

[51] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[52] M. K. Johnson and H. Farid, "Detecting photographic composites of people," in *International Workshop on Digital Watermarking*. Springer, 2007, pp. 19–33.

[53] ——, "Exposing digital forgeries through specular highlights on the eye," in *International Workshop on Information Hiding*. Springer, 2007, pp. 311–325.

[54] M. Kirchner, "Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue," in *proc. of the ACM Workshop on Multimedia and Security*. ACM, 2008, pp. 11–20.

[55] 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*, vol. 5, no. 4, pp. 883–892, December 2010.

[56] I. Kokkinos and A. Yuille, "Scale invariance without scale selection," *Department of Statistics, UCLA*, 2011.

[57] N. Komodakis and G. Tziritas, "Image completion using efficient belief propagation via priority scheduling and dynamic pruning," *IEEE Transactions on Image Processing*, vol. 16, no. 11, pp. 2649–2661, 2007.

[58] S. Korman and S. Avidan, "Coherency sensitive hashing," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1607–1614.

[59] M. Kutner, C. Nachtsheim, J. Neter, and W. Li, *Applied Linear Statistical Models*. McGraw-Hill, 2004.

[60] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," in *ACM Transactions on Graphics (ToG)*, vol. 22, no. 3. ACM, 2003, pp. 277–286.

[61] D. Labartino, T. Bianchi, A. D. 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 *IEEE International Workshop on Multimedia Signal Processing*, 2013, pp. 494–499.

[62] A. Langille and M. Gong, "An efficient match-based duplication detection algorithm," in *Canadian Conf. on Computer and Robot Vision*, 2006.

[63] L. Li, S. Li, H. Zhu, and X. Wub, "Detecting copy-move forgery under affine transforms for image forensics," *Computers & Electrical Engineering*, vol. 40, no. 6, pp. 1951–1962, 2014.

[64] Y. Li, "Image copy-move forgery detection based on polar cosine transform and approximate nearest neighbor searching," *Forensic Science International*, vol. 224, no. 1-3, pp. 59–67, 2013.

[65] S. X. Liao and M. Pawlak, "On the accuracy of zernike moments for image analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1358–1364, December 1998.

[66] S.-Y. Liao and T.-Q. Huang, "Video copy-move forgery detection and localization based on Tamura texture features," in *International Congress on Image and Signal Processing*, 2013, pp. 864–868.

[67] C. Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, M. L. Miller, and Y. M. Lui, "Rotation, scale,and translation resilient watermarking for images," *IEEE Transactions on Image Processing*, vol. 10, pp. 767–782, 2001.

[68] C.-S. Lin and J.-J. Tsay, "A passive approach for effective detection and localization of region-level video forgery with spatio-temporal coherence analysis," *Digital Investigation*, vol. 11, no. 2, pp. 120–140, 2014.

[69] Z. Lin, J. He, X. Tang, and C.-K. Tang, "Fast, automatic and fine-grained tampered jpeg image detection via dct coefficient analysis," *Pattern Recognition*, vol. 42, no. 11, pp. 2492–2501, 2009.

[70] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[71] J. Lukáš, J. Fridrich, and M. Goljan, "Detecting digital image forgeries using sensor pattern noise," in *proc. of the SPIE*, vol. 6072, 2006, pp. 720Y–11.

[72] B. Mahdian and S. Saic, "Detection of copy-move forgery using a method based on blur moment invariants," *Forensic Science International*, vol. 171, pp. 180–189, 2007.

[73] 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*, vol. 1, December 2012.

[74] N. Mondaini, R. Caldelli, A. Piva, M. Barni, and V. Cappellini, "Detection of malevolent changes in digital video for forensic applications," in *Proc. of SPIE Conference on Security, Steganography and Watermarking of Multimedia*, vol. 6505, 2007.

[75] G. Muhammada, M. Hussain, and G. Bebis, "Passive copy move image forgery detection using undecimated dyadic wavelet transform," *Digital Investigation*, vol. 9, pp. 49–57, 2012.

[76] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[77] P. Mullan, D. Cozzolino, L. Verdoliva, and C. Riess, "Residual-based Forensic Comparison of Video Sequences," in *Internation Conference on Image Processing, Proceedings*, I. ICIP, Ed., Beijing, 2017, pp. 1–6.

[78] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[79] I. Olonetsky and S. Avidan, "Treecann-kd tree coherence approximate nearest neighbor algorithm," *Computer Vision–ECCV 2012*, pp. 602–615, 2012.

[80] X. Pan and S. Lyu, "Region duplication detection using image feature matching," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 857–867, Dec. 2010.

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

[82] K. A. Patwardhan, G. Sapiro, and M. Bertalmío, "Video inpainting under constrained camera motion," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 545–553, 2007.

[83] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *ACM Transactions on graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 313–318.

[84] A. Piva, "An overview on image forensics," *ISNR Signal Processing*, pp. 1–22, Oct. 2012.

[85] A. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images," *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3948–3959, Oct. 2005.

[86] H. Ravi, A. Subramanyam, G. Gupta, and B. A. Kumar, "Compression noise based video forgery detection," in *IEEE International Conference on Image Processing*, 2014, pp. 5352–5356.

[87] 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)*, vol. 43, no. 4, p. 26, 2011.

[88] S. Ryu, M. Lee, and H. Lee, "Detection of copy-rotate-move forgery using zernike moments," in *Information Hiding Conference*, 2010, pp. 51 – 65.

[89] S.-J. Ryu, M. Kirchner, M.-J. Lee, and H.-K. Leee, "Rotation invariant localization of duplicated image regions based on zernike moments," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1355–1370, Aug. 2013.

[90] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1. IEEE, 2003, pp. I–I.

[91] S. Sharma and S. V. Dhavale, "A review of passive forensic techniques for detection of copy-move attacks on digital videos," in *Advanced Computing and Communication Systems (ICACCS), 2016 3rd International Conference on*, vol. 1. IEEE, 2016, pp. 1–6.

[92] Y. Shen, F. Lu, X. Cao, and H. Foroosh, "Video completion for perspective camera under constrained motion," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3. IEEE, 2006, pp. 63–66.

[93] Y. Sheng and H. Arsenault, "Experiments on pattern recognition using invariant fourier mellin descriptors," *J. Opt. Soc. Amer.*, vol. 3, pp. 771–776, 1986.

[94] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[95] R. D. Singh and N. Aggarwal, "Video content authentication techniques: a comprehensive survey," *Multimedia Systems*, pp. 1–30, 2017.

[96] K. Sitara and B. M. Mehtre, "Digital video tampering detection: An overview of passive techniques," *Digital Investigation*, vol. 18, pp. 8–22, 2016.

[97] M. C. Stamm, M. Wu, and K. R. Liu, "Information forensics: An overview of the first decade," *IEEE Access*, vol. 1, pp. 167–200, 2013.

[98] M. Stamm, W. Lin, and K. R. Liu, "Temporal forensics and anti-forensics for motion compensated video," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1315–1329, August 2012.

[99] 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*, 2009, pp. 1–4.

[100] A. Subramanyam and S. Emmanuel, "Video forgery detection using HOG features and compression properties," in *IEEE International Workshop on Multimedia Signal Processing*, 2012, pp. 89–94.

[101] T. Sun, W. Wang, and X. Jiang, "Exposing video forgeries by detecting MPEG double compression," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2012, pp. 1389–1392.

[102] Y. Sutcu, B. Coskun, H. Sencar, and N. Memon, "Tamper detection based on regularity of wavelet transform coefficients," in *IEEE International Conference on Image Processing (ICIP)*, vol. 1, Sep. 2007, pp. 397–400.

[103] M. Teague, "Image analysis via the general theory of moments," *Journal of the Optical Society of America*, vol. 70, no. 8, pp. 920–930, Aug. 1980.

[104] J. Thies, M. Zollhöfer, M. Niessner, L. Valgaerts, M. Stamminger, and C. Theobalt, "Real-time expression transfer for facial reenactment." *ACM Trans. Graph.*, vol. 34, no. 6, pp. 183–1, 2015.

[105] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Niessner, "Face2face: Real-time face capture and reenactment of rgb videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2387–2395.

[106] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 5, pp. 815–830, 2010.

[107] L. Verdoliva, D. Cozzolino, and G. Poggi, "A reliable order-statistics-based approximate nearest neighbor search algorithm," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 237–250, January 2017.

[108] ——, "A feature-based approach for image tampering detection and localization," in *Information Forensics and Security (WIFS), 2014 IEEE International Workshop on*. IEEE, 2014, pp. 149–154.

[109] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *arXiv preprint arXiv:1408.2927*, 2014.

[110] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting double MPEG compression," in *ACM Workshop on Multimedia and Security*, 2006, pp. 37–47.

[111] ——, "Exposing digital forgeries in video by detecting duplication," in *ACM Multimedia and Security Workshop*, 2007, pp. 35–42.

[112] ——, "Exposing digital forgeries in video by detecting double quantization," in *ACM Workshop on Multimedia and Security*, 2009, pp. 39–48.

[113] J. Wickramasuriya, M. Alhazzazi, M. Datt, S. Mehrotra, and N. Venkatasubramanian, "Privacy-protecting video surveillance," in *SPIE Int.l Symposium on Electronic Imaging*, 2005, pp. 64–75.

[114] Q. Wu, S. Wang, and X. Zhang, "Log-polar based scheme for revealing duplicated regions in digital images," *IEEE Signal Processing Letters*, vol. 18, no. 10, pp. 559–652, 2011.

[115] Y. Wu, X. Jiang, T. Sun, and W. Wang, "Exposing video inter-frame forgery based on velocity field consistency," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 2674–2678.

[116] Y. Xin, M. Pawlak, and S. Liao, "Accurate computation of zernike moments in polar coordinates," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 581–587, Feb. 2007.

[117] P.-T. Yap, X. Jiang, and A. Kot, "Two-dimensional polar harmonic transforms for invariant image representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1259–1270, July 2010.

[118] I. Yerushalmy and H. Hel-Or, "Digital image forgery detection based on lens and sensor aberration," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 71–91, 2011.

[119] F. Zernike, "Diffraction theory of the knife-edge test and its improved form, the phase-contrast method," *Monthly Notices of the Royal Astronomical Society*, vol. 94, pp. 377–384, 1934.

[120] J. Zhao and J. Guo, "Passive forensics for copy-move image forgery using a method based on dct and svd," *Forensic Science International*, vol. 233, no. 1-3, pp. 158–166, 2013.

[121] J. Zhao and W. Zhao, "Passive forensics for region duplication image forgery based on harris feature points and local binary patterns," *Mathematical Problems in Engineering*, vol. 2013, p. 12 pages, 2013.