https://doi.org/10.2352/ISSN.2470-1173.2021.4.MWSF-337

Video Source Identification from MP4 Data Based on Field Values in Atom/Box Attributes

Erik Gelbing, Leon Würsching, Sascha Zmudzinski, Martin Steinebach
Fraunhofer Institute for Secure Information Technology SIT, Darmstadt, Germany

Abstract

Identifying the source of a video recording created by a camera or smartphone has been a common and challenging task in media forensics for many years. We present an approach for source identification on the very common MP4 file format. In extension to related works, we propose to consider the suitability of attribute field values and their respective order in the atom/box tree in a specific manner. The significance of a field attribute and its particular value for source identification will be reflected by means of up and down weighting during the training and the matching process. Experimental result indicate that our approach allows distinguishing major brands. Even device identification is possible for a subset of our training data.

Introduction

Nowadays we can observe an ever increasing number of digital video data being recorded, edited and eventually being archived and/or circulating on the Internet. Such video data can provide valuable digital evidence to forensic experts. One interesting task in video forensic investigations is *camera source identification*: reconstructing the life cycle of digital video in terms of the original device, its brand or model, or the codec software involved.

Especially for user-created content the video source often is a smartphone or a consumer digital camera. Here, widely used storage formats are the family of MP4-like multimedia containers like **.mp4, *.mov* and **.3gp*. Hence it will be in the focus of our work.

One effective approach for camera identification from MP4-like digital data is analyzing its meta data. The general approach is as follows: Let x be a query video of unknown origin (referred to as "camera" or "device") and $\mathscr{C}$ the set of origins considered (for simplicity we assume we know that $\mathscr{C}$ in fact contains the origin of x). The task is to find the correct origin of x by analyzing its semantical and syntactical metadata and comparing it to the available data for each origin $c \in \mathscr{C}$.

There are two decisions to make in order to implement an analyzer following this approach:

- How to compute a score for each origin considered, expressing the similarity of the query video x to other videos from mentioned origin?
- How to interpret these scores classify the most likely origin?

How these aspects have been implemented in the related work and in our contribution is explained in the following.

Outline of This Work At first, we will recap the basics of the MP4 file format family. Then we will give an overview about the related work and on open research questions in the field of MP4 camera identification. We will then explain our proposed approach and provide experimental results. Finally we will conclude by discussing and summarizing our results and by proposing further research directions.

MP4 Basics

A common industry standard for storing encoded video and/or audio data is the *MP4* multimedia container format. It is specified as an international standard [1],[8] as an extension of the Apple Quicktime (MOV) format [9]. The MP4 format features a hierarchical tree-like data structure divided into *boxes* (also denoted as *atoms*). The box type is indicated by a four byte ASCII identifier. Boxes (and their nested sub-boxes, resp.) contain the encoded audio/video data (in the `mdat` box) and its meta data in the `moov` box and other box types. The data values inside these boxes are organized in attribute *fields*.

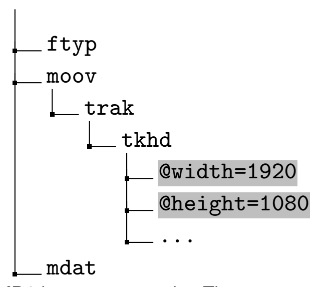


Figure 1. *MP4 box tree example: The `@width/@height` fields and their respective values inside the `tkhd` sub-box)*

For example, the value of the video frames' pixel size is stored in the track header box (`tkhd`) in its `@width` and `@height` fields. Their full path descriptor in the box tree is e.g. `moov/trak/tkhd/@width=1920`, see Fig. 1.

The field values can feature syntactical information for decoding and rather semantical information about the content useful to human users.

Apart from the explicit information in the field values, the box tree provides *implicit* forensic information by the *order* in which boxes are present and also by which optional boxes are present *at all*. Hence, some authors in video forensics even consider the *order* of the sub-boxes [2],[3]. For this, the path descriptors carry a suffix that indicates the absolute index number[1] of the sub box in its respective (parent) box, i.e. in the above example `/moov1/trak1/tkhd0/@width=1920`

[1]As common tradition for computer scientists, we start numbering at zero.

Related Work

In MP4-like containers such as MP4, MOV or 3GP files, there are a lot of traces available for source camera identification. The audio and video stream can be analyzed e.g. using CFA de-mosaicing [17], PRNU-based sensor fingerprinting [19], ENF-based audio forensic [18], and by many more approaches in multimedia forensics.

Alternatively, traces can be derived from *meta data* entries. Early works were designed for MP3 audio e.g. by *Böhme et al.* [10] or for JPEG still images [11] by *Gloe et al.* rather than container formats for multimedia. Similar to the latter for JPEG, a work by *Hall* [12, 13] presents an approach focused on the path structure of the MP4 sub-boxes. By contrast, a work by Güera, Delp et al. [14] analyzes "stream descriptors" from selected field *values* by means of different classifiers (random forests, SVM).

Works by *Iuliani, Piva et al.* [2], *Yang, Piva et al.* [15] or *López et al* [3]. analyze *both* the path and the value of box fields. In addition, *Rudnikovich et al.* [16] present a parsing and visualization tool that facilitates manual file inspection. Many of these works apply or at least discuss their approaches for both source identification and integrity verification.

Earlier Works by Iuliani, Yang, Piva et. al

Basis of our work is an approach by *Iuliani et al.* [2]. The authors state that a video x can be completely described by the set of occurring field-value pairs (also referred to as attributes) $\Omega_X = \omega_1, \ldots, \omega_m$ [2] and their corresponding paths $p_X(\omega_i)$.

In order to compute a score denoting the chances x has been created with camera c, they analyze at each attribute in x and calculate a percentage of occurrences from similar field-value pairs with equal path in c. The authors use this percentage to compute a likelihood ratio $L_c(x)$ denoting the probability that x was created from a camera of type c The resulting scores are normalized by applying log and authors propose to identify an origin for x in $\mathscr{C}$ whenever $L_c(x) > 0$.

That work was extended for analyzing video files that undergo video editing or exchanging files in social media platforms [15].

Proposed Approach

In this section, we introduce an approach to identify the source camera for MP4-like containers by analysis of the presence and ordering of syntactical meta data entries in MP4 boxes. In light of the related work we introduce three novel features to further improve the identification task:

- *Block list.* Some boxes include fields that have distinct values for virtually every video file. Other boxes contain information that is unspecific for the source camera. None of these box types contribute to source camera identification and thus, they should be canceled from the equation.
- *Path interpretation.* There are several ways to interpret the position of a box within the box tree. All existing approaches that considers the ordering of boxes have difficulties with the insertion of boxes. Therefore, we introduce a new index-based path type that is robust against the insertion of boxes.
- *Weighted matching.* Not all fields are relevant for source camera identification. While matching, we consider how distinctive a field is for a given source camera. That is, fields that show the same value for all videos of that camera are considered more distinctive than fields that show multiple values. [3]

[2] Actually, they consider the list of occurring attributes, so there may be attributes $\omega_i, \omega_j \in \Omega_X$ such that $i \neq j \wedge \omega_i = \omega_j \wedge p_X(\omega_i) = p_X(\omega_j)$. However, while evaluating scores, they reduce the weight of repeated fields, which results in the same score as considering a set in the first place.

Notation

Let x be a video that is analyzed for source camera identification. Let $c \in \mathscr{C}$ be a source camera from the set of all source cameras and let C be the set of cameras that are considered for identifying the source of x. Let $\bar{X}_c$ be a set of videos that are known to have been recorded with c.

Each video x contains a list of N_x boxes $B_x := (b_0, \ldots, b_{N_x-1})$ with boxes $b_i \in \mathscr{B}$ from the set of all boxes. As the boxes are actually structured in a tree, we use a function ρ to map the box b to its path within a video x.

Each box b contains a set of M_b fields $\mu(b) := \{f_0, \ldots, f_{M_b-1}\}$. As each field may show a different value for each video, $\vartheta_x(f)$ maps a field f to the value it shows in the video x. For simplicity, let $\omega := (f, \vartheta_x(f))$ denote a field-value if the video x can be derived from context. Furthermore, let $F_x := \bigcup_{b \in B_x} \{(f, \vartheta_x(f)) | f \in \mu(b)\}$ denote the set of field-values in a video x.

In the remainder of this section, we will discuss the principle steps of our algorithm including an in-depth discussion of the novel features we introduce. Our algorithm can be split into three steps:

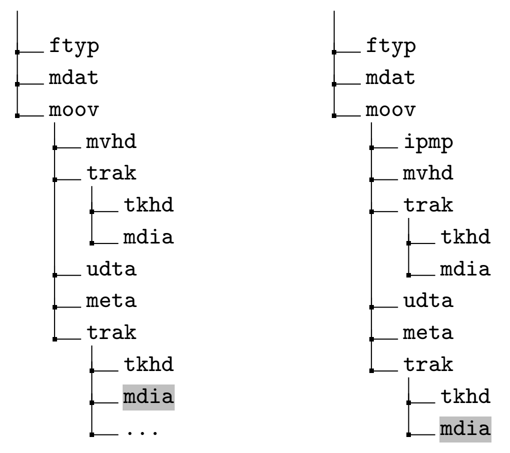
- *Static Analysis.* The relevance of each box is classified with respect to source identification. From this classification a block list is derived.
- *Training Stage.* MP4 videos with known source camera are used to train our model. For each source camera, the model learns which boxes are present in which ordering.
- *Identification.* For a given test video, the presence and position of each box is checked against the trained model. If there is a significant number of matches for a source camera (ignoring boxes from the block list), chances are high that the test video has been recorded with that camera.

Static Analysis

The overall goal of our algorithm is to identify the source camera of a test video. Each test video includes a box tree where each box consists of one or multiple *field-value pairs* (field-values). To identify the source camera of a test video, we observe each field-value of the test video and search for videos that have an *equal* field-value. The three properties of a field-value are (1) the field-key, (2) the value and (3) the position within the box tree. Two field-values are equal if and only if they are equal in all three properties.

However, there are field-values that cannot virtually ever be equal for two non-identical videos. In order to cancel such field-values from the equation, we implement a block list.

[3] Note how after applying the block list, we can assume that we have already dealt with two cases: (1) fields that show unique values for every video and (2) fields that show the same value for every video.



(a) Box tree B1 (b) Box tree B2

Figure 2. *Example: Box trees MP4 files from different devices. The second `mdia` box at different path positions*

Block list

There are various reasons why field-values cannot contribute to the distinction of the source cameras: On the one hand, there are fields which values can be expected to be specific for this (and only this), such as video file, such as timestamps. On the other hand, there are fields which values seem to have little to no purpose for identification, e.g. geolocations.

Therefore, a *block list* is derived with boxes that contain field-values with little or no significance for source camera identification. Unfortunately, this step requires manual consultation of the MP4 specifications [1],[8] to learn the intention of each field. In the identification stage of our algorithm, the block list will be queried to remove boxes which seem to have lots of irrelevant field-values completely from the matching equation.

Training Stage

The MP4 specification describes a lot of boxes for MP4-like containers, some of which are optional. Furthermore, it does not specify box ordering for the most part, which leaves camera manufacturers with the freedom to decide which optional boxes are included and how they are positioned.

We want to take advantage of the fact that each camera manufacturers has to decide how the box tree is structured and what (optional) boxes are included. Therefore, we need to learn what boxes each manufacturer includes and how they build the box tree.

One crucial property of field-values is their position within the box tree. Related work [2],[3] introduces different path types. In the following, we discuss pitfalls of existing path types and derive a novel path type that is used in our algorithm.

Path Interpretation Types

Boxes in MP4-like containers are arranged in a tree, which suggests to define their positions with a path representation. Figure 2 shows a section of two typical box trees. To illustrate different approaches to the interpretation of paths, we focus on the highlighted *media* box on the bottom of the figure.

The basic idea is similar to the file path in a directory structure. A box path is a list of all boxes that are traversed when going from the root to the box. We refer to this type of paths as *unsorted path order index* (path **type 1**). The type 1 path for our example box is `moov/trak/media`. While this is a very straightforward approach to define paths, a closer look reveals that the path is not unambiguous for the box tree from Figure 2a (left) as two boxes qualify for the same path.

This motivates to consider indices when referring to boxes on the path. The *absolute path order index* (path **type 2**) implements indices by numbering the children of each (parent) box. The type 2 path for our example box is `/moov2/trak1/mdia1`. In contrast to type 1 paths, type 2 paths unambiguously reference boxes. However, the approach is not robust to the insertion of boxes.

This can have severe consequences in the identification stage, as the following example shows: Assume different file's box tree as depicted in Figure 2b (right), that is equal the box tree from 2a but with an additional `ipmp` box as the first child of `moov2`. As a result, the indices of all boxes subsequent to the added box will be shifted by one.

Alternatively, we introduce the *insert-robust path order index* (path **type 3**). In contrast to path type 1 where index i means that the box is the i-th child of its parent box, path type 3 considers the *occurrences* of a box type relative to its parent box. In path type 3, index i means that the box is the i-th child of its parent box *with the same box type*.

Hence, the type 3 path representation is denoted as `/moov_0/trak_1/mdia_0` for *both* examples[4] in Figure 2.

The key advantage of type 3 paths is that they are more robust against the insertion of boxes. Note how the box tree from Figure 2b is the same as 2a but with an additional box. The type 2 path for our example box changes, the type 3 path for our example box is the same in both box trees. As a result, path type 3 solves the ambiguity problem of path type 1 and the insertion vulnerability of path type 2.

Identification Stage

We identify the source camera of a test video x by calculating a score $\phi_x(c)$ for every camera c. Therefore, we prepare a score list Φ where the score for each camera is stored as a tuple $(c, \phi_x(c))$.

Weighted Matching

We evaluate each field-value individually considering how much that field-value ω_i contributes to the decision if the test video x matches camera c. As the weighting factor, we define the *diversity* κ that has the following properties: Let $F_c := \bigcup_{\bar{x} \in \bar{X}_c} F_{\bar{x}}$ be the list of field-values contained in a camera c.

If there is no field-value in F_c that has the same path as ω, then $\kappa_c(\omega) := 0$.

Otherwise, we consider three factors:

- the number of available source videos from camera c and
- the number of distinct values that occur for the given field in any video of camera c.
- the number of videos from camera c, in which the given field exists with equal path and equal value.

[4]Kindly note the underscore_notation for type 3 paths

The first factor is the cardinality of the set of videos from source c: $|\bar{X}_c|$.

For the second factor, we filter $|\bar{X}_c|$ for videos that contain an equal field-value[5]:

$$\sigma_c(\omega) := |\{\bar{x} \in \bar{X}_c | \exists \bar{\omega} \in \bar{x} : \omega = \bar{\omega}\}| \quad . \tag{1}$$

The last factor is constructed by

$$K_c(\omega) := \bigcup_{(\bar{f}, \vartheta_{\bar{x}}(\bar{f})) \in F_c} \{\vartheta_{\bar{x}}(f) | \bar{f} = f \wedge \rho_x(\bar{f}) = \rho_x(f)\} \tag{2}$$

where $\omega = (f, \vartheta_x(f))$.

Finally, we define the *diversity* $\kappa_c(\omega)$ as

$$\kappa_c(\omega) := \begin{cases} 0 & \text{if } |K_c(\omega)| = 0 \\ \frac{|\bar{X}_c|}{|K_c(\omega)|} & \text{otherwise} \end{cases} \quad . \tag{3}$$

In simple words, the proposed diversity property describes how many different outcomes of a particular field-value are observed in the training data. A high diversity value indicates that this field-value is not very specific across files from the same device and is hence only little suitable for matching.

Finally, the similarity score is calculated by applying the following steps for every camera c:

1. Generate the list F_x of all fields contained in the video x.
2. With $\omega = (f, \vartheta_x(f))$, calculate

$$\phi_\omega(c) := \sum_{\omega \in F_x} \varepsilon(f) \cdot \sigma_c(\omega) \kappa_c(\omega) \quad .$$

3. Add $(c, \phi_x(c))$ to Φ.

The camera with the highest score is most likely to be the source camera of the test video. Therefore, the output of the identification stage is the camera $\hat{c}$ with maximum score

$$\phi_x(\hat{c}) = \max_{(c, \phi_x(c)) \in \Phi} \phi_x(c). \tag{4}$$

Evaluation

For the task of *Device Classification*, we determine Φ for a test-video x. The goal is to evaluate whether the camera with the highest score $\hat{c}$ does in fact represent the device x has been created with (*TOP1*). Further we kept track of whether the classification for a video does improve when we also consider if the correct device was identified within the two highest (*TOP2*) as well as the three highest (*TOP3*) score. Although these additional scores come at the cost of a high false positive rate, they help understanding the approaches ability to enclose the origin of x. If multiple cameras achieve the same score within these TOP's, we report on a classification for all of them. We further aggregate the results from the Device Classification to receive *Brand Classification*. Therefore we acknowledge a correct classification whenever the cameras brand (e.g. Apple) is identified correctly.

Datasets

We evaluate our approach on the ACID[5],[6],[20] dataset[6] as well as on the VISION[4] dataset[7], see Table 1.

Table 1: Datasets in our evaluation

Dataset	#brands	#cams.	#vids.	min. #vids. per cam.
ACID [5],[6]	15	29	10,069	187
VISION [4]	10	32	561	10

Evaluation Approach

Since the ACID dataset already provides a split into *train* and *evaluation* data[8], we also use this split during our test. We evaluate our approach for all videos from the *evaluation* dataset, by comparing each of them to 187 randomly selected videos per camera from the *train* dataset. Please note that we imported the train data beforehand. This means after randomly selecting the videos once, all further comparisons are calculated on the same data.

For each test video we determine its Φ and collect the *TOP1, TOP2* and *TOP3* classification. After testing all videos, the resulting data can be used to determine a device classification confusion matrix.

We carry out the evaluation on the VISION dataset identically. Since the dataset does not provide a split into train and test data, we perform our test on a subset of 10 randomly selected videos for each camera. Out of this subset, each video is compared against the selected data (excluding itself). This means the video is classified for all 9 videos in the subset of its own camera, as well as against 9 randomly selected videos of each remaining device in the subset. This is, because the amount of videos per device should be equal during the test, and a video cannot be part of the test data while it is being evaluated.

Different total numbers of training files per device (187 for ACID, 10 for VISION, resp.) is given by the different volume of testdata available and (for the test on the VISION data) to have sufficiently many files left for the comparison stage.

Computational Effort

Due to the static analysis of the available data, the computational effort for a device classification of a video using our proposed approach, is extremely low. Our implementation classifies all 725 videos (duration 6.0 seconds per file in average, i.e. approx. 1.2 hours total) of the ACID evaluation dataset in 5.3 seconds if the data per video has been parsed and imported to the database previously. Importing the subset of 5423 videos (duration approx. 9.0 hours) we use from the train dataset with the *MP4 Parser*[7] library takes 32.6 seconds.

[5]The equality of field-values ($\omega = \bar{\omega}$) means that both field-values are equal in their field, value and path: $f = \bar{f} \wedge \vartheta_x(f) = \vartheta_{\bar{x}}(\bar{f}) \wedge \rho_x(f) = \rho_{\bar{x}}(\bar{f})$ where $\omega = (f, \vartheta_x(f)), \bar{\omega} = \vartheta_{\bar{x}}(\bar{f}))$

[6]We had to remove the devices M14, M21, M25, M26 and M33 since they are not .mp4 or .mov files. We also had to remove M09 since the MP4 parser was not able to correctly parse it.

[7]We removed D07, D22 and D35 due to since the MP4 Parser was not able to correctly parse them.

[8]9344 train, 725 evaluation

Table 2: VISION data – Brand Classification – Confusion Table Test 1

	Apple	Asus	Huawei	LG	Microsoft	OnePlus	Samsung	Sony	Wiko	Xiaomi	TOP2	TOP3
Apple	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
Asus	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
Huawei	0.00	0.00	**0.88**	0.00	0.00	0.04	0.00	0.00	0.00	0.10	1.00	1.00
LG	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
Microsoft	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	1.00	1.00
OnePlus	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	1.00	1.00
Samsung	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	1.00	1.00
Sony	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	1.00	1.00
Wiko	0.00	0.00	0.00	0.00	0.00	0.30	0.00	0.00	**0.80**	0.30	0.90	1.00
Xiaomi	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**	1.00	1.00

Table 3: Proposed Block List

`mdat`	`mvhd`	`tkhd`	`mdhd`	`stsc`	`stsz`	`stz2`
`stco`	`co64`	`hdlr`	`tref`	`stts`	`ctts`	`padb`
`subs`	`hmhd`	`cmhd`	`smhd`			

Table 4: Evaluation Results Overview - Top 1

Test	Dataset	Blocklist	Path type	Brand AUC	Device AUC
1	VISION	no	2	**1.0000**	**0.9793**
2	VISION	no	3	**1.0000**	0.9635
3	VISION	yes	2	0.9656	0.9030
4	VISION	yes	3	0.9717	0.9029
5	ACID	no	2	0.9995	0.9274
6	ACID	no	3	**1.0000**	0.9291
7	ACID	yes	2	0.9643	0.9163
8	ACID	yes	3	**1.0000**	**0.9342**

Evaluation Results

We evaluate under the usage of absolute path order indices (type 2) and insertion-robust path order indices (type 3) as well as with the block list (as proposed in Table 3) enabled and disabled, resp.

Table 2 shows results of the relative frequency of test devices that were matched to the training data from different brands (VISION dataset). Large values for *correct* matches (on the diagonal mostly 1.0) and small values for wrong matches (mostly 0.0 off the diagonal) demonstrate the promising matching performance.

Let us assume that we have a set of N test files from which we know that the were recorded with the same unknown device (or brand, resp.). Then the above relative frequencies can be evaluated for defining reasonable decision thresholds: how many of those N files should indicated a positive match in order to assign *all* files in the set to a device or brand? That is analyzed in terms of receiver operating characteristic in Fig. 3 and Table 4.

Table 4 provides an overview of classification performance for the used datasets for the *TOP1* classification. Overall the evaluation shows very good results on brand classification with an AUC above 0.95 for all tests. Even the more specific device classification still shows an AUC over 0.90 on all tests.

The results are indifferent across the two datasets with regards to the proposed block list and path type. Notice that for brand classification, we are able to achieve an AUC of 1.0 on both datasets.

Nevertheless, for the more recent camera devices from the ACID data, using the block list and insertion-invariant path representations (type) appears to be beneficial for brand and device identification in terms of AUC values. Hence we suggest the usage of insert-robust paths (type 3) from these results, due to their benefits on possible changes on the MP4 box structure of certain devices in the future. However, refining the block list is still proposed to be a challenging task which presumes a deep knowledge of the MP4/Quicktime Specification [1],[8],[9].

Summary

In this paper we proposed an approach to source identification of MP4-like container files. This algorithm bases on existing work by [2] and introduces three novel features: A *block list* that cancels irrelevant field-values from the equation, a novel *path type* that is robust against the insertion of boxes while combining the advantages of existing path types and *weighted matching* that considers a field-values diversity among videos of a camera.

Our evaluation shows that we achieve very good classification performance for brand identification on the ACID and VISION test datasets. The proposed extensions to the related work are beneficial when evaluated on the ACID Training set from 2019. In extension to related works we also address the aspect of identifying the individual devices instead of its brand.

As future work, we propose to refine the block list to consider *fields* instead of boxes. This can be beneficial for source camera identification as there are cases where one box combines both relevant and irrelevant fields.

References

[1] ISO/IEC, 14496-12:2015 – Information technology, Coding of Audio-visual Objects, Part 12: ISO Base Media File Format, Technical Committee ISO/IEC JTC 1/SC 29, ISO/IEC, Geneva, Switzerland, (2015).

[2] M. Iuliani, D. Shullani, M. Fontani, S. Meucci and A. Piva, "A Video Forensic Framework for the Unsupervised Analysis of MP4-Like File

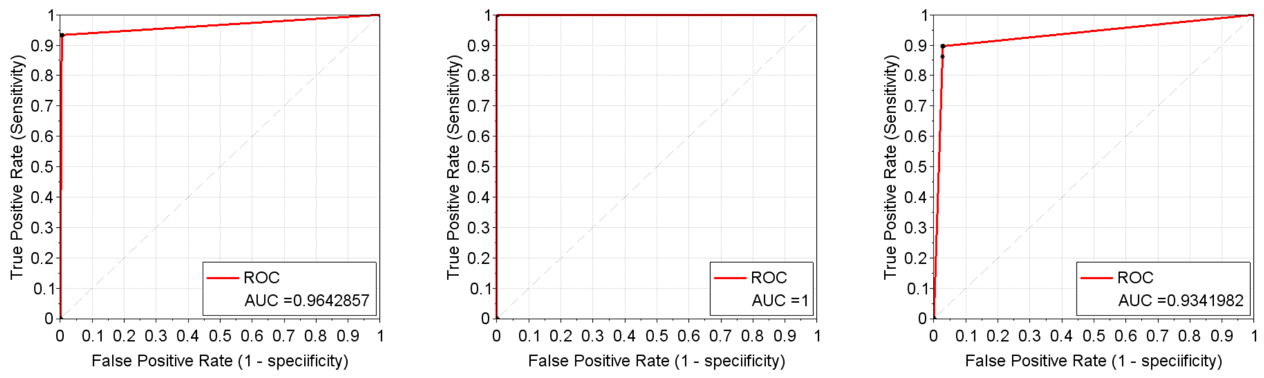


Figure 3. *Receiver Operating Characteristics for Different Setups (ACID dataset)– left: brand identification Test 7; middle: brand identification Test 8; right: device identification Test 8*

Container," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 635-645 (2019).

[3] R. Ramos López, E. Almaraz Luengo, A. L. Sandoval Orozco and L. J. G. Villalba, "Digital Video Source Identification Based on Container's Structure Analysis," in IEEE Access, vol. 8, pp. 36363-36375 (2020).

[4] Shullani, D., Fontani, M., Iuliani, M., Al Shaya, O., and Piva, A. (2017). VISION: a video and image dataset for source identification. EURASIP Journal on Information Security, article 15 (2017)

[5] B. Hosler, O. Mayer, X. Zhao, C. Chen, J. Shackleford, M. Stamm. (2019). Video-ACID. IEEE Dataport. https://dx.doi.org/10.21227/4dzb-h573

[6] B. C. Hosler, X. Zhao, O. Mayer, C. Chen, J. A. Shackleford and M. C. Stamm, The Video Authentication and Camera Identification Database: A New Database for Video Forensics", IEEE Access, vol. 7, pp. 76937-76948 (2019)

[7] Java MP4 Parser. Accessed: Feb. 7, 2021. [Online]. Available: http://www.github.com/sannies/mp4parser

[8] ISO/IEC, 14496-14:2020 – Information technology, Coding of Audio-visual Objects, Part 12: ISO Base Media File Format, edition 3.0, ISO/IEC, Geneva, Switzerland (2020).

[9] Apple Computer Inc., QuickTime File Format (2002).

[10] R. Böhme and A. Westfeld. Statistical characterisation of MP3 encoders for steganalysis. Proc. ACM Workshop on Multimedia and Security, pg. 25-34 (2004).

[11] T. Gloe, A. Fischer, M. Kirchner, Forensic analysis of video file formats, Digital Investigation, vol. 11, suppl. 1, pg. S68-S76 (2014).

[12] J. Randolph Hall, MPEG-4 Video Authentication Using File Structure and Metadata, thesis, University of Colorado, USA (2015).

[13] J. Hall, MP4 Video Authentication Using File Structure and Metadata, DFWRS USA 2015 (2015).

[14] D. Güera, S. Baireddy, P. Bestagini, S. Tubaro, E. J. Delp, We Need No Pixels: Video Manipulation Detection Using Stream Descriptors, CoRR/arXiv (2019).

[15] P. Yang, D. Baracchi, M. Iuliani, D. Shullani, R. Ni, Y. Zhao, A. Piva, Efficient Video Integrity Analysis Through Container Characterization, IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 5, pp. 947-954 (2020).

[16] A. S. Rudnikovich, K. A. Rylov, Methods for Graphic Visualization of Video File Structure and Determining the Origin in Forensics, Proc. Int. Multi-Conf. on Engineering, Computer and Information Sciences (SIBIRCON), pg. 0436-0441 (2019).

[17] M. Kirchner, Efficient estimation of CFA pattern configuration in digital camera images, Proc. SPIE 7541, Media Forensics and Security II, 754111 (2010).

[18] C. Grigoras, Applications of ENF Criterion in Forensic Audio, Video, Computer and Telecommunication Analysis, Forensic Science International, Volume 167, Issues 2-3, pg. 136-145 (2007).

[19] T. Filler, J. Fridrich and M. Goljan, Using Sensor Pattern Noise for Camera Model Identification, Proc. IEEE International Conference on Image Processing, pg. 1296-1299 (2008).

[20] Video ACID: A database for the study of video forensic algorithms, Multimedia & Information Security Lab, College of Engineering, Drexel University, Accessed: Feb. 15, 2021. [Online]. Available: https://misl.ece.drexel.edu/video-acid/ .

Acknowledgment

This work was supported by the German Federal Ministry of Education and Research (BMBF) and/or the and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of "ATHENE – Disinformation on Corona" and "Lernlabor Cybersicherheit".

Author Biography

Erik Gelbing is a bachelor student of Computer Science at Technische Universität Darmstadt, Germany, and student researcher at the Media Security and IT Forensics division at Fraunhofer SIT.

Leon Würsching is a master student of IT Security at Technische Universität Darmstadt, Germany. He works as an assistant researcher at the Media Security and IT Forensics division at Fraunhofer SIT.

Sascha Zmudzinski is a senior researcher at Fraunhofer SIT in the Media Security and IT Forensics division. He received his PhD at the TU Darmstadt in 2017 for his work on authentication audio watermarking.

Martin Steinebach is the Manager of the Media Security and IT Forensics division at Fraunhofer SIT. In 2003 he received his PhD at the TU Darmstadt for this work on digital audio watermarking. In 2016 he became honorary professor at the TU Darmstadt.

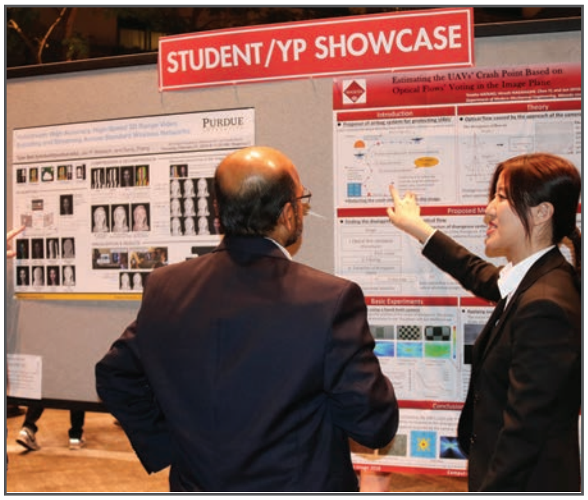
JOIN US AT THE NEXT EI!

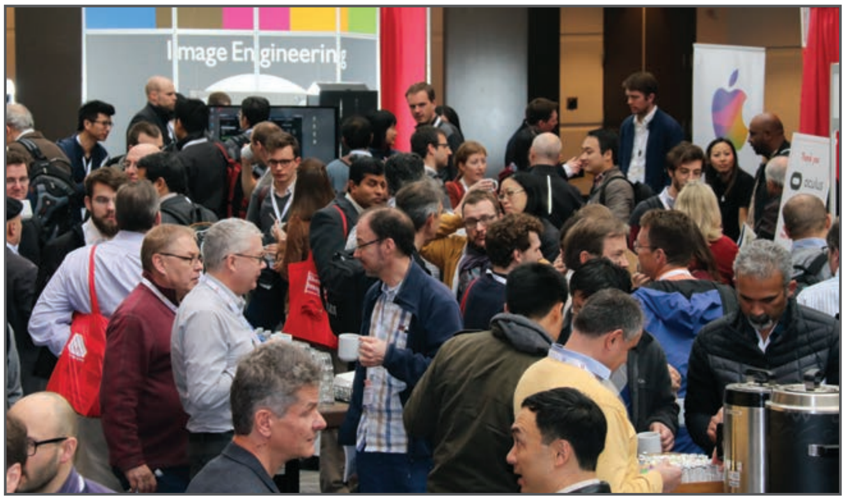
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!


STUDENT/YP SHOWCASE
Estimating the UAVs' Crash Point Based on Optical Flows' Voting in the Image Plane
PURDUE



AR is a fundamental and transformative interface and platform.

20+ years from my initial working demo to today's AR displays

Optimistic that it won't take another 20 to reach ubiquitous consumer AR!

• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org