REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
17-10-2022	Final Report	1-Jul-2017 - 31-Oct-2020

4. TITLE AND SUBTITLE

Final Report: High Performance Techniques to Identify the Source and Authenticity of Digital Videos Using Multimedia Forensics

5a. CONTRACT NUMBER
W911NF-17-2-0087

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES

Drexel University
Office of Research
3201 Arch Street, Suite 100
Philadelphia, PA 19104 -2875

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
70824-BB-RIF.6

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Matthew Stamm
UU	UU	UU	UU		19b. TELEPHONE NUMBER 215-895-5894

RPPR Final Report
as of 17-Oct-2022

Agency Code: 21XD

Proposal Number: 70824BBRIF **Agreement Number: W911NF-17-2-0087**
INVESTIGATOR(S):

Name: Matthew Stamm
Email: MStamm@coe.drexel.edu
Phone Number: 2158955894
Principal: Y

Organization: **Drexel University**
Address: Office of Research, Philadelphia, PA 191042875
Country: USA
DUNS Number: 002604817 EIN: 231352630

Report Date: 30-Nov-2020 Date Received: 17-Oct-2022
Final Report for Period Beginning 01-Jul-2017 and Ending 31-Oct-2020
Title: High Performance Techniques to Identify the Source and Authenticity of Digital Videos Using Multimedia Forensics
Begin Performance Period: 01-Jul-2017 **End Performance Period:** 31-Oct-2020
Report Term: 0-Other

Submitted By: Matthew Stamm Email: MStamm@coe.drexel.edu
Phone: (215) 895-5894

Distribution Statement: 1-Approved for public release; distribution is unlimited.

STEM Degrees: 5 **STEM Participants:** 10

Major Goals: 1.1 Project Goals
This project will address the following tasks aimed at developing technologies to identify the source and authenticity of digital videos using multimedia forensics:

Task1 - Building forensic algorithms that identify traces specific to the make and model of the camera that captured a video.

Task 2 - Developing algorithms that fuse multiple forensic traces and identify the source of a digital video.

Task 3 - Establishing algorithms to determine the authenticity of digital videos by detecting evidence of video forgery.

Task 4 - Developing field deployable command line and GUI tools implementing the algorithms developed in Tasks 1–3 that leverage multi-core architectures for acceleration.

Task 5 - Performing ground truth video data collection, along with testing and validation of all software and algorithms developed in Tasks 1-4.

For more details, please refer to the uploaded PDF document.

Accomplishments: Please refer to the uploaded PDF document.

Training Opportunities: Please refer to the uploaded PDF document.

Results Dissemination: Please refer to the uploaded PDF document.

Honors and Awards: Please refer to the uploaded PDF document.

Protocol Activity Status:

Technology Transfer: Please refer to the uploaded PDF document.

RPPR Final Report
as of 17-Oct-2022

PARTICIPANTS:

Participant Type: PD/PI
Participant: Matthew Stamm
Person Months Worked: 15.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Co PD/PI
Participant: Nagarajan Kandasamy
Person Months Worked: 8.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Co PD/PI
Participant: James Shackleford
Person Months Worked: 4.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Xinwei Zhao
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Belhassen Bayar
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Tai Nguyen
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Hunter Kippen
Person Months Worked: 4.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Michael Spanier
Person Months Worked: 3.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Shengbang Fang
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Chen Chen
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Brian Hosler
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Owen Mayer
Person Months Worked: 12.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Undergraduate Student
Participant: Jacob Baron
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
National Academy Member: N

ARTICLES:

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: IEEE Access
Publication Identifier Type: DOI Publication Identifier: 10.1109/ACCESS.2019.2922145
Volume: 7 Issue: First Page #: 76937
Date Submitted: 10/17/22 12:00AM Date Published:
Publication Location:
Article Title: The Video Authentication and Camera Identification Database: A New Database for Video Forensics
Authors: Brian C. Hosler, Xinwei Zhao, Owen Mayer, Chen Chen, James A. Shackleford, Matthew C. Stamm
Keywords: Forensics, multimedia databases, benchmark testing, video signal processing
Abstract: Modern technologies have made the capture and sharing of digital video commonplace; the combination of modern smartphones, cloud storage, and social media platforms have enabled video to become a primary source of information for many people and institutions. As a result, it is important to be able to verify the authenticity and source of this information, including identifying the source camera model that captured it. While a variety of forensic techniques have been developed for digital images, less research has been conducted toward the forensic analysis of videos. In part, this is due to a lack of standard digital video databases, which are necessary to develop and evaluate state-of-the-art video forensic algorithms. In this paper, to address this need, we present the video authentication and camera identification (video-ACID) database, a large collection of videos specifically collected for the development of camera model identification algorithms. The video-ACID database contains
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

CONFERENCE PAPERS:

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
Date Received: 17-Oct-2022 Conference Date: 04-May-2020 Date Published:
Conference Location: Barcelona, Spain
Paper Title: Open Set Video Camera Model Verification
Authors: Owen Mayer, Brian Hosler, Matthew C. Stamm
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
Date Received: 17-Oct-2022 Conference Date: 12-May-2019 Date Published:
Conference Location: Brighton, United Kingdom
Paper Title: A Video Camera Model Identification System Using Deep Learning and Fusion
Authors: Brian Hosler, Owen Mayer, Belhassen Bayar, Xinwei Zhao, Chen Chen, James A. Shackleford, Matthew
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 0-Other
Conference Name: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)
Date Received: 17-Oct-2022 Conference Date: 14-Jun-2020 Date Published:
Conference Location: Seattle, WA, USA
Paper Title: Detecting Video Speed Manipulation
Authors: Brian C. Hosler, Matthew C. Stamm
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 17-Oct-2022

Partners

,

I certify that the information in the report is complete and accurate:
Signature: Matthew C. Stamm
Signature Date: 10/17/22 12:38AM

FINAL REPORT

Submitted to:
The Army Research Office (ARO)
Defense Forensic Science Center (DFSC)

Agreement Administered By:
Office of Naval Research Regional Office (Chicago)

Federal Grant Number
W911NF-17-2-0087

Project Title
High Performance Techniques to Identify the Source and Authenticity of Digital Videos Using Multimedia Forensics

Prepared & Submitted By
Dr. Matthew C. Stamm, Project PI
Elec. & Comp. Eng. Dept., Drexel University
3141 Chestnut Street
Philadelphia, PA 19104
Phone: (215) 895 5894
Email: mstamm@drexel.edu

Submission Date
22 April 2021

DUNS and EIN Numbers
DUNS Number: 002604817
EIN Number: 23-1352630

Recipient Organization
Drexel University
3201 Arch Street, Suite 100
Philadelphia, PA 19104

Project Period
01 July 2017–31 October 2021

End Date of Reporting Period
30 October 2020

Report Term
FINAL

Signature of Submitting Official

Contents

1 Accomplishments

1.1 Project Goals

This project addressed the following tasks aimed at developing technologies to identify the source and authenticity of digital videos using multimedia forensics:

Task 1 Build forensic algorithms that identify traces specific to the make and model of the camera that captured a video.

Task 2 Develop algorithms that fuse multiple forensic traces and identify the source of a digital video.

Task 3 Establish algorithms to determine the authenticity of digital videos by detecting evidence of video forgery.

Task 4 Develop a field deployable command line and graphical user interface (GUI) tools implementing the algorithms developed in Tasks 1–3 that leverage multi-core architectures for acceleration.

Task 5 Perform ground truth video data collection, along with testing and validation of all software and algorithms developed in Tasks 1-4.

Milestones and Deliverables

This program provided deliverables in 6 month increments over a period of 24 months. These deliverables and their associated schedule of delivery are listed below. In addition to these deliverables, we will provide monthly execution reports delivered within 30 days of the end of each month, as well as a final report upon 100% fund expenditure.

Deliverables Provided 6 Months After Contract Award

- Baseline techniques to identify and extract forensic traces specific to the make and model of a video's source camera from a single video frame.
- Initial algorithm to determine the make and model of a video's source camera using basic fusion of these forensic traces.
- Algorithm to identify video forgeries by identifying LCA anomalies in individual video frames.
- Intel/AMD compatible software implementation of these algorithms using Python, or if necessary, Matlab.
- Initial software tool development for extracting prediction errors and motion vectors from digital video files.
- Testing and validation on videos from 10 different camera models.

Deliverables Provided 12 Months After Contract Award

- Improved techniques to extract forensic traces for camera model identification from within one video frame and from across multiple frames.
- Updated algorithm to determine the make and model of an video's source device using fusion of these forensic traces.
- Initial algorithm to detect video frame and segment deletion.
- Algorithm to detect video forgery by comparing LCA anomalies across multiple frames.
- Implementation of 6 month algorithmic deliverables into a high-performance software tool compatible with Intel/AMD based systems.
- Testing and validation on videos collected from 20 different camera models.

Deliverables Provided 18 Months After Contract Award

- Improved techniques to extract source camera traces from across single and multiple video frames.
- Improved algorithm capable of providing confidence scores when fusing forensic traces to determine the make and model of a video's source camera.
- Updated algorithms to detect video frame deletion and LCA-based evidence of video forgery with improved accuracy and computational efficiency.
- Implementation of 12 month algorithmic deliverables into high-performance software tool compatible with Intel/AMD based systems with no Matlab dependence.
- Initial cross-platform user interface for forensic software tool.
- Testing and validation on videos collected from 35 different camera models.

Deliverables Provided 24 Months After Contract Award

- High performance implementation of source camera feature extraction algorithms.
- Final, fully trained implementation of video camera model identification algorithm.
- High performance Intel/AMD compatible implementation of LCA-based forgery detection and frame deletion detection forensic algorithms.
- Fully implemented software tool with cross-platform graphical user interface supporting Linux, OS X, and Windows; user documentation; and source code.
- Testing and validation on full video database.

Time Line Chart by Task

Table 1 provides a summary breakdown of the project timeline, organized in terms of the various proposed tasks.

Task	Description	Phase 1	Phase 2	Phase 3	Phase 4
#1	Single frame traces	✓	✓	✓	
	Multi-frame and codec based traces		✓	✓	✓
	Other forensic traces			✓	✓
#2	Hierarchical Decision Fusion Framework	✓	✓	✓	
	Machine Learning Approaches for Feature Fusion		✓	✓	✓
	Camera Identification Confidence Scores				✓
#3	Detecting Forgeries using Lateral Chromatic Aberration	✓	✓	✓	✓
	Detecting Video Segment Deletion or Addition			✓	✓
#4	Research implementations in Python	✓	✓	✓	✓
	High performance CPU/GPU implementations		✓	✓	✓
	User Interface Development		✓	✓	✓
	Software documentation				✓
#5	Collection of videos for our testing library.	✓	✓	✓	✓
	Validation of the video collection.	✓	✓	✓	✓

Table 1: Project schedule and milestones.

1.2 What was accomplished under these goals?

Summary:

Below is a summary of the major accomplishments achieved under this project.

- We developed a deep learning based approach to extract forensic traces that can be used to identify a video's source camera model from patches of size 256×256 pixels of a video frame.

- We developed a fusion system to combine forensic traces extracted from multiple regions throughout a video and perform highly accurate source camera model identification.
    - Experimental results show that this system is able to identify a video's source camera model with an **average accurace of 96.0%**.

- We implemented an algorithm to perform camera model identification using information stored in an video file's header and container, such as quantization tables, group of picture structure, frame resolution, and file format.
    - This algorithm is capable of identifying a small set of possible source camera models without making classification errors.
    - This algorithm is designed to be integrated into our hierarchical decision fusion framework.

- We designed a new hierarchical decision fusion framework to combine multiple traces in order to perform highly accurate camera model identification.
    - This framework integrates each of the algorithms described above.
    - Experimental results show it can perform camera model identification with an **average accuracy of 98.42%**.

- We developed a new algorithm to perform open set source camera model verification (i.e. camera model identification when not all possible sources are known). This algorithm compares two videos and is able to determine if they were captured by the same camera model, even if the algorithm has never encountered these camera models before.
    - This algorithm can determine if two videos were captured by the same camera model with an **average accuracy of 96.0%** if they were encountered during training.
    - This algorithm can still determine if two videos were captured by the same camera model with an average accuracy of **average accuracy of 87.3%** even of the sources of these two videos were never previously encountered.

- We **discovered new forensic traces introduced into a video file when its speed is manipulated**, i.e. the video is either slowed down or sped up the video.

- Using the video speed manipulation traces that we discovered, we **developed new algorithms to detect video speed manipulation** and to estimate the rate by which a video's speed has been modified.

- We developed an algorithm to identify fake content within a digital video by detecting inconsistencies in a videos' lateral chromatic aberration (LCA).

- We collected a large scale video database for testing and validation purposes. Our database contains approximately 12,173 images collected from 36 different camera models (using 46 unique devices).
- We created **a software package titled the *Video Authentication and Source Identification Toolkit*** containing implementations of all algorithms developed under this project.
  - This software package can be run on individual videos or large data sets from either a graphical user interface or from the command line.
  - We created a 59 page user manual describing how to install our software, use it to identify the source camera model of images under investigation, and how to train it to recognize new camera models.

Background: Overview of a Camera's Internal Video Processing Pipeline

The set of physical components and processing algorithms that a device uses to capture a digital video is known as it's video processing pipeline. The video processing pipeline of a typical digital camera is shown in Fig. 1. Light enters a camera by first passing through a lens. Since most sensors are only capable of measuring one color of light at each pixel location, the light next passes through a color filter array (CFA), which is an optical filter that allows only one color of light (red, green, or blue) to hit the sensor at a particular pixel location. The sensor then measures the light intensity of the corresponding color band at each pixel location, and produces an image constructed of three partially sampled color layers. Next, the unobserved color values at each pixel location are interpolated using color values at nearby pixel locations that were directly observed. This process is known as demosaicing. After demosaicing, the video often undergoes post-processing, such as white balancing and color correction, followed by compression (such as MPEG-4, H.264, etc.) before the final video file is output or stored.

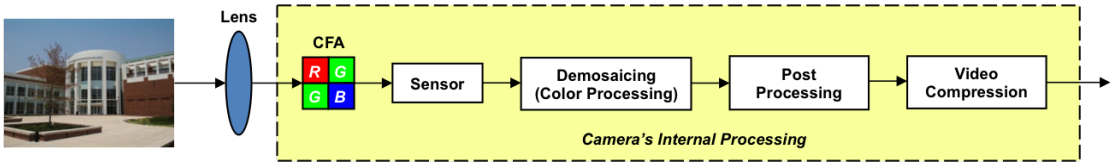


Figure 1: Video processing pipeline of a typical digital camera.

While the video processing pipelines of different devices are generally composed of common elements (e.g. lens, sensor, color processing algorithms, image compression, etc.), the specific implementation and characteristics of each element typically varies from manufacturer to manufacturer and from model to model. The video processing pipeline is consistent, however, across all devices that share a common model and manufacturer.

1.2.1 Deep Learning Camera Model Identification

We developed a deep learning based approach to automatically identify and extract forensic traces left in a video by its source camera model.

This algorithm uses a specially designed convolutional neural network (CNN) to extract camera model traces from video. The CNN can be used to directly perform camera model identification,

or these traces can be provided to a higher-level fusion algorithm. Furthermore, this CNN is able to automatically learn forensic traces associated with a new camera model directly from training data.

The development and implementation of this algorithm addressed Tasks 1 and 3 of this project.

Algorithm Overview

This algorithm operates by analyzing and extracting forensic traces from a small $N \times N$ pixel window of a video frame. We refer to this small window as a 'patch'. To extract camera model traces, we developed a modified version of the MISLnet CNN architecture developed in our previous research [1, 2]. MISLnet's architecture is shown below in Fig. 2. It is composed of a constrained convolutional layer of three filters, followed by 5 convolution blocks (convolution, batch normalization, activation, then max pooling), then two fully-connected blocks (matrix multiplication followed by activation), and finally, a fully-connected layer. Full implementation details can be found in [1].

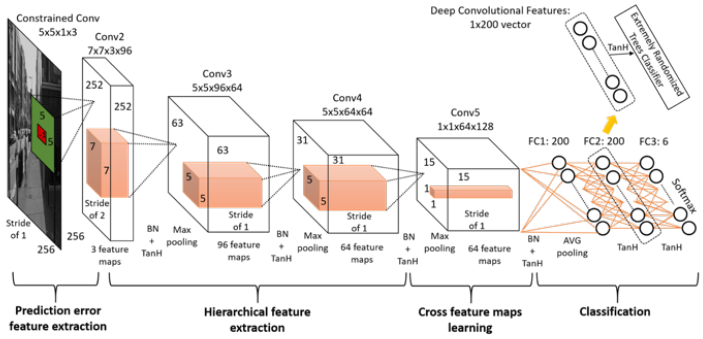


Figure 2: The architecture of our MISLnet CNN that was used to extract source camera model traces from videos.

To adapt this CNN for video camera model identification, we made several modifications to its architecture. First, we chose an input patch size of 256×256 pixels. To adapt to color patches, we removed the constraint on the first convolutional layer, and increased the number of first layer filters from 3 to 6. We then assigned one neuron in the last fully connected layer to each possible camera model in the training set. During training, a softmax activiation is placed upon the final layer of neurons in our CNN. This normalizes each neuron's activation across all of the neurons in the final layer. As a result, the softmax activated value of the k^{th} neuron in the final output layer of our CNN can be interepreted as the probability that the video was captured by the k^{th} camera (each camera in the training set is assigned an index number). Additionally, while our CNN can be used to analyze any video frame, its performance is maximized when I-frames are analyzed. Performance drops when P-frames (or B-frames) are used.

After it has been trained, our CNN accepts a patch from a video frame as an input, then produces a vector ϕ of the unnormalized neuron activations in its final layer an output. The vector ϕ contains a representation of the source camera model traces present in the patch under analysis. This vector can then be passed to our fusion algorithm (described in Section **??**). Alternatively, this vector can be used directly for camera model identification by choosing source camera model as the one corresponding to the entry of ϕ with the highest activation. As described above, a like-

lihood that this is the correct camera model can be obtained by passing ϕ through the softmax function.

A more detailed description of this system can be found in [3].

Testing and Validation Results

We conducted an experiment to evaluate the performance of our forensic CNN. In this experiment, we measured the ability of our CNN to perform camera model identification using only a single patch from a video (i.e. no fusion of multiple traces from the same video was performed).

To do this, we used an experimental dataset of videos taken by 36 different camera models. Each camera model was used to capture on average roughly 250 videos, which were split into separate training and testing sets. More details of this dataset are provided in Section 1.2.8. To train our CNN, we selected 10,000 patches from each camera model for a total of 360,000 training patches. We then evaluated the our CNN's classification accuracy using 1,000 evaluation patches from each camera model,for a total of 36,000 evaluation patches.

In this experiment, our system achieved an overall camera model identification accuracy of 81.7% using only a single patch from each video. In Table 2 we show the average accuracy of our CNN per-camera when performing identification using only a single patch. Results are further broken down for videos encoded only in the H.264 format (middle column) and for videos that encoded in either the H.264 or the less common MJPEG format (right column).

From this these results, we can see that when only using traces in single patch our CNN is still able to achieve a strong source camera model identification accuracy. This demonstrates that our CNN is able to extract forensic traces that contain a significant amount of information about a video's source. We note that typically, our CNN would not be used to perform camera model identification on the basis of only one patch. It is designed to be used in conjunction with the system described in Section 1.2.2 which fuses traces from multiple patches throughout a single video file. This fusion substantially boosts the overall accuracy when performing source camera model identification.

1.2.2 Patch-Level Trace Fusion System For Camera Model Identification

We created a system to combine patch-level forensic traces extracted by the system described in Section 1.2.1 and produce video-level forensic decisions. This system rapidly produces highly accurate camera model identification results along with likelihood scores for all potential source camera models.

The development and implementation of this algorithm addresses Tasks 2 & 4 of this project.

Algorithm Overview

Our fusion system operates by combining forensic traces extracted from multiple patches within a frame and multiple patches across frames to make a single forensic decision.

The first step of our system chooses video locations for forensic trace extraction. Our research has shown that best performance is achieved by analyzing spatially and temporally diverse patches taken from I-frames throughout the video. As a result, our system randomly selects N patches per frame from F randomly chosen I-frames throughout the video. Forensic trace vectors $\boldsymbol{\phi}_k$ are then extracted from each patch using the system described in Section 1.2.1

Next, these forensic traces are fused together to form a single forensic trace vector. Three fusion

Table 2: Single-patch per-class accuracy of video camera model identification. Results are displayed for only video encoded using H.264, as well as video encoded using H.264 and MJPEG.

Camera Model	H.264 Only	H.264 and MJPEG
M00_Apple_iPhone_8_plus	93.9%	85.0%
M01_Asus_Zenfone_3_Laser	76.9%	78.3%
M02_Canon_EOS_SL1	84.2%	92.3%
M03_Canon_EOS_T6i	96.7%	98.0%
M04_Canon_Powershot_SX530_HS	77.5%	74.6%
M05_Canon_Powershot_SX610_HS	89.3%	85.8%
M06_Canon_VIXIA_HF_R800	98.9%	96.7%
M07_Fujifilm_Finepix_S8600		67.7%
M08_Fujifilm_Finepix_XP80	92.2%	96.9%
M09_GoPro_Hero_Session	98.4%	96.6%
M10_Google_Pixel_1	69.1%	79.1%
M11_Google_Pixel_2	85.3%	92.9%
M12_Huawei_Honor_6X	69.8%	73.4%
M13_Huawei_Mate_SE	81.9%	79.5%
M14_JVC_EverioR	95.3%	92.1%
M15_Kodak_Ektra	60.8%	80.0%
M16_LG_Q6	62.4%	71.3%
M17_LG_X_Charge	66.3%	83.9%
M18_Moto_E4	74.6%	62.9%
M19_Moto_G5_Plus	48.8%	54.6%
M20_Nikon_Coolpix_S33	81.1%	93.9%
M21_Nikon_Coolpix_S3700		95.1%
M22_Nikon_Coolpix_S7000	96.5%	95.5%
M23_Nokia_6.1	68.9%	70.3%
M24_Olympus_Stylus_Tough_TG-860	86.5%	95.3%
M25_Panasonic_FZ200	96.7%	93.7%
M26_Panasonic_HC-V180	96.2%	90.8%
M27_Samsung_Galaxy_J7_Pro	71.3%	77.4%
M28_Samsung_Galaxy_S3	75.6%	86.0%
M29_Samsung_Galaxy_S5	53.5%	40.2%
M30_Samsung_Galaxy_S7	64.2%	72.5%
M31_Samsung_Galaxy_Tab_A	67.3%	62.9%
M32_Samsung_J5-6	73.7%	68.9%
M33_Sony_Cybershot_DSC-WX350	89.0%	95.7%
M34_Sony_Xperia_L1	73.4%	61.2%
M35_Yi_4k_Action_Camera	90.8%	98.6%

approaches were investigated while designing our system. Each approach can be interpreted as a voting scheme in which a patch casts a vote for the source camera model.

The first fusion approach f_1 computes the sum of each forensic trace vector $\boldsymbol{\phi}_k$ after being passed through the softmax function. This can be interpreted as a fractional voting scheme, in which each patch gets to cast a portion of a vote for a source camera model. Since the softmax function normalizes the sum of all entries of $\boldsymbol{\phi}_k$ to 1, this fusion still restricts each patch to cast one vote in total. The second fusion approach f_2 is majority vote scheme. Here, each patch casts a vote for the source camera corresponding to the entry of $\boldsymbol{\phi}_k$ with the highest activation. The third fusion approach f_3 directly combines all of the forensic trace vectors without normalization. This allows each patch to vote unequally for all camera models. Each patch may vote for each camera model, positively or negatively, with as much weight as it wants. With this voting scheme, patches can have negative weight against classes they want to reject, and are able to indicate how confident they are in one or more class with respect to other patches.

Our final system utilizes the direct fusion scheme f_3, since experimental validation (described below) showed that this approach yields the highest performance. After all forensic trace vectors have been fused, our system chooses the source camera model as the one corresponding to the entry of the fused forensic trace vector with the highest value. Relative likelihoods of each possible source camera are provided by passing the fused forensic trace vector through the softmax function.

A more detailed description of this system can be found in [3].

Testing and Validation Results

We conducted several experiments to validate the performance of our fusion system. These are described below.

Evaluation of Different Trace Fusion Rules

First, we evaluated the performance of our system when using each of the three fusion rules considered: f_1 - fractional voting, f_2 - majority vote, and f_3 unnormalized trace fusion. To do this, we created an experimental dataset from videos captured by 20 cameras in the full dataset described in Section **??**. Over 250 videos of at least 5 seconds in duration were collected from each camera model.

We divided the videos from each camera model into separate training sets and testing sets, resulting in 5,962 total training videos and 650 total testing videos. A database of 320,880 I-frame patches was created from the set of training videos by selecting three I-frames from each video, dividing those frames into 256×256 patches, and labeling each patch with the true camera model. This process was repeated to create a dataset of 35,280 I-frame patches from the set of testing videos. We then used our fusion system to identify the source camera model of each video while using each different fusion rule.

Table 3: Single frame accuracy of tested fusion techniques

Fusion function	f_1	f_2	f_3
Accuracy	91.0%	92.1%	92.6%

We fused the forensic traces extracted from all 28 non-overlapping patch in one randomly selected I-frame from each of the 650 videos in the testing set, then computed the average camera

model identification accuracy achieved with each fusion rule. The accuracies achieved for each fusion rule are shown in Table 3. From this table, we can see that fusion rule f_3, i.e. the unnormalized fusion of all trace vectors, yielded the highest performance. This suggests that forcing all patches to have the same number of votes is worse than allowing each patch to vote according to it's own relative confidence. We note that our chosen approach, f_3, outperforms the commonly used majority vote strategy, described by f_2

Large Scale Evaluation of our Trace Fusion System

Next, we performed a large scale evaluation of our system's performance utilizing fusion rule f_3. To conduct this experiment, we utilized videos from all 36 camera models in our experimental dataset described in Section 1.2.8. Each camera model was used to capture on average roughly 250 videos, which were split into separate training and testing sets. More details of this dataset are provided in Section 1.2.8. We performed patch-level trace extraction using the trained CNN-based forensic trace extractor described in the Testing and Validation results subsection of Section 1.2.1. We then evaluated our system's source camera model identification accuracy on the 925 videos in the testing set. When performing camera model identification, our system fused $N = 3$ patches per frame taken from $F = 3$ I-frames for a total of 9 patches per video.

Our system achieved an overall camera model identification accuracy of 96.0% on the testing set. This corresponds to an increased performance of 14.3 percentage points over using the results obtained using our CNN to perform camera model identification on the basis of a single patch (see Section 1.2.1). Furthermore, Table 4 shows identification accuracy of our fusion system for each camera model. Results are further broken down for videos encoded only as H.264 and for videos that encoded in either the H.264 or the less common MJPEG format. When compared with the single-patch identification results displayed in Table 2, we can see that our fusion system provides considerable performance improvements for identifying the source of videos captured several camera models.

These results demonstrate that when our fusion system is used to combine the traces extracted from multiple patches throughout a single video, we can perform highly accurate source camera model identification for digital videos.

1.2.3 Video Header and File Container Traces

We implemented an algorithm to extract forensically significant traces from an video's header and file container. These traces are then used to eliminate camera models that do not produce videos with an identical set of video compression header traces and identify a small set of possible source camera models. This algorithm was designed to be implemented into our data fusion framework described in Section 1.2.4.

The development and implementation of this algorithm addressed Tasks 1, 2, and 4 of this project.

Algorithm Overview

An video file's header and file container contains several forensically significant traces. The container is the file format that structures the video stream, i.e. *'.mp4', '.mov', '.h264'*, etc. The header is the portion of the video file containing information about how the video is decoded and parsed.

Table 4: Per-class fusion accuracy of video camera model identification.

Camera Model	H.264 Only	H.264 and MJPEG
M00_Apple_iPhone_8_plus	100%	100%
M01_Asus_Zenfone_3_Laser	100%	96%
M02_Canon_EOS_SL1	100%	100%
M03_Canon_EOS_T6i	100%	100%
M04_Canon_Powershot_SX530_HS	88%	92%
M05_Canon_Powershot_SX610_HS	100%	100%
M06_Canon_VIXIA_HF_R800	100%	100%
M07_Fujifilm_Finepix_S8600		84%
M08_Fujifilm_Finepix_XP80	100%	100%
M09_GoPro_Hero_Session	100%	100%
M10_Google_Pixel_1	96%	100%
M11_Google_Pixel_2	100%	100%
M12_Huawei_Honor_6X	92%	96%
M13_Huawei_Mate_SE	96%	100%
M14_JVC_EverioR	100%	100%
M15_Kodak_Ektra	84%	100%
M16_LG_Q6	92%	96%
M17_LG_X_Charge	88%	100%
M18_Moto_E4	88%	96%
M19_Moto_G5_Plus	72%	72%
M20_Nikon_Coolpix_S33	100%	100%
M21_Nikon_Coolpix_S3700		98%
M22_Nikon_Coolpix_S7000	100%	100%
M23_Nokia_6.1	100%	100%
M24_Olympus_Stylus_Tough_TG-860	96%	92%
M25_Panasonic_FZ200	100%	100%
M26_Panasonic_HC-V180	100%	100%
M27_Samsung_Galaxy_J7_Pro	100%	96%
M28_Samsung_Galaxy_S3	100%	96%
M29_Samsung_Galaxy_S5	92%	64%
M30_Samsung_Galaxy_S7	96%	92%
M31_Samsung_Galaxy_Tab_A	92%	88%
M32_Samsung_J5-6	96%	96%
M33_Sony_Cybershot_DSC-WX350	100%	100%
M34_Sony_Xperia_L1	92%	76%
M35_Yi_4k_Action_Camera	100%	100%

In this project, we utilized several different pieces of information to determine from the video's header and container, including video format, discrete cosine transform (DCT) quantization tables for each frame type, the video's resolution (i.e. 720p, 1080p, etc.), the group-of-picture (GOP) structure, and color space information. Since video compression has a significant impact on the quality of a video, most digital camera manufacturers design their own proprietary DCT quantization tables. Additionally, most camera models only capture video with a particular resolution, produce videos with only one file container format, and often utilize only a small set of possible GOP structures.

While the values these traces are not typically unique to a particular camera model, very few camera models will produce an identical set of traces. We use these traces to sort camera models into groups called equivalence classes. All camera models in an equivalence class are capable of producing an identical set of video file header and container traces.

We developed a software module to extract these traces from a video video file and compare them to a hash table of precomputed equivalence classes. This module can be trained learn a new reference table of equivalence classes by extracting traces from videos with known sources, then grouping camera models with identical traces.

This module can operate on its own and provide an investigator with the equivlaence class of camera models with identical traces to those present in a video under investigation. However, it is designed primarily to be used in conjunction with the multi-trace fusion framework described in Section 1.2.4.

1.2.4 Hierarchical Multi-Trace Fusion Framework For Camera Model Identification

We developed a hierarchical data fusion framework to combine information gathered from a video's header and container with forensic information extracted using our CNN-based system. This system increases the overall camera model identification accuracy by leveraging information provided by all forensic traces available.

The development and implementation of this framework addressed Tasks 2 & 4 of this project.

Algorithm Overview

Our trace fusion framework is designed to exploit intrinsic information hierarchies that naturally occur within camera model identification fingerprints. For example, video file header and container traces can be used to eliminate possible source camera models without running the risk of making decision errors, but they typically cannot be used to identify an individual source camera model.

Our framework uses a tree-based decision structure that exploits these latent information hierarchies. Early stages of the framework eliminate camera models that cannot possibly be the video's source camera. Later stages identify a single source camera model from the reduced set of possible camera models. This divide-and-conquer approach reduces the number of camera models that later stages need to choose from, thus increasing the accuracy of classifiers used at these stages. Since these early stage decisions are error-free, this reduces the overall error rate of the entire framework.

A diagram of our system is shown in Figure. 3. An overview of of our decision fusion algorithm is described below:

1. Video file header and container information is extracted using the system described in Section 1.2.3.

2. These traces are used to sort a video into one of N equivalence classes. Each equivalence class is the set of all camera models able to produce images with the same video file header and container information.
3. Our CNN-based system for extracting forensic traces from video patches described in Section 1.2.1 is used to extract traces from several patches throughout the video.
4. The trace fusion system described in Section 1.2.2 is used to combine the traces extracted in Step 3, then use these fused traces distinguish between the possible source camera models in the equivalence class identified in Step 2.

Testing and Validation Results

We conducted an experiment to validate the performance of our hierarchical trace fusion system. To measure the performance gains over our patch-level fusion system, we repeated the large scale validation experiment performed in Section 1.2.2, but this time we used our full hierarchical trace fusion system. To accomplish this, we used the patch-level training data created for the experiments in Section 1.2.1 to train unique CNN feature extractors for each equivalence class. We then evaluated the performance of our hierarchical trace fusion system on the testing dataset of videos described in Section 1.2.8. This set was the same set of 925 videos captured by 36 different camera models used to evaluate our patch-level fusion system in Section 1.2.2.

Our hierarchical fusion system achieved a source camera model identification accuracy of 98.42%. This very high accuracy demonstrates that when all available traces our fused, our system can achieve very strong camera model identification performance.

When using only our patch-level fusion framework, our system achieved a 96.0% accuracy. As a result, our hierarchical fusion framework yields a 2.42% increase in accuracy. While initially, this improvement may appear small, it should be noted that this corresponds to a relative error reduction of 60.5% since the system could only achieve a maximum of a 4.0% increase in performance. Relative error reduction is defined as the percentage of the total error that was reduced by a system improvement.

1.2.5 Open Set Camera Model Verification

The systems described in Sections 1.2.1-1.2.4 are camera model identification approaches in which a video's source camera is chosen from a set of known potential sources. In some cases, however, an investigator may not know the set of all possible source camera models that could have captured a video. This is referred to in machine learning literature as an "open set" classification problem - i.e. the set of all possible choices is not completely known to the classifier. If the "closed set" classification approach is used for source camera model identification, the classifier will always choose a known camera model from the training set, even if the true source is from an unknown camera model.

To address this issue, we developed an open set camera model identification algorithm. This algorithm is able to compare forensic traces in a video under investigation to those in a reference video, then determine if these videos were captured by the same source camera model. This approach can accurately match a video's source camera to with known camera models, or it can reject all models and tell the investigator that the video originated from an unknown model. Fur-

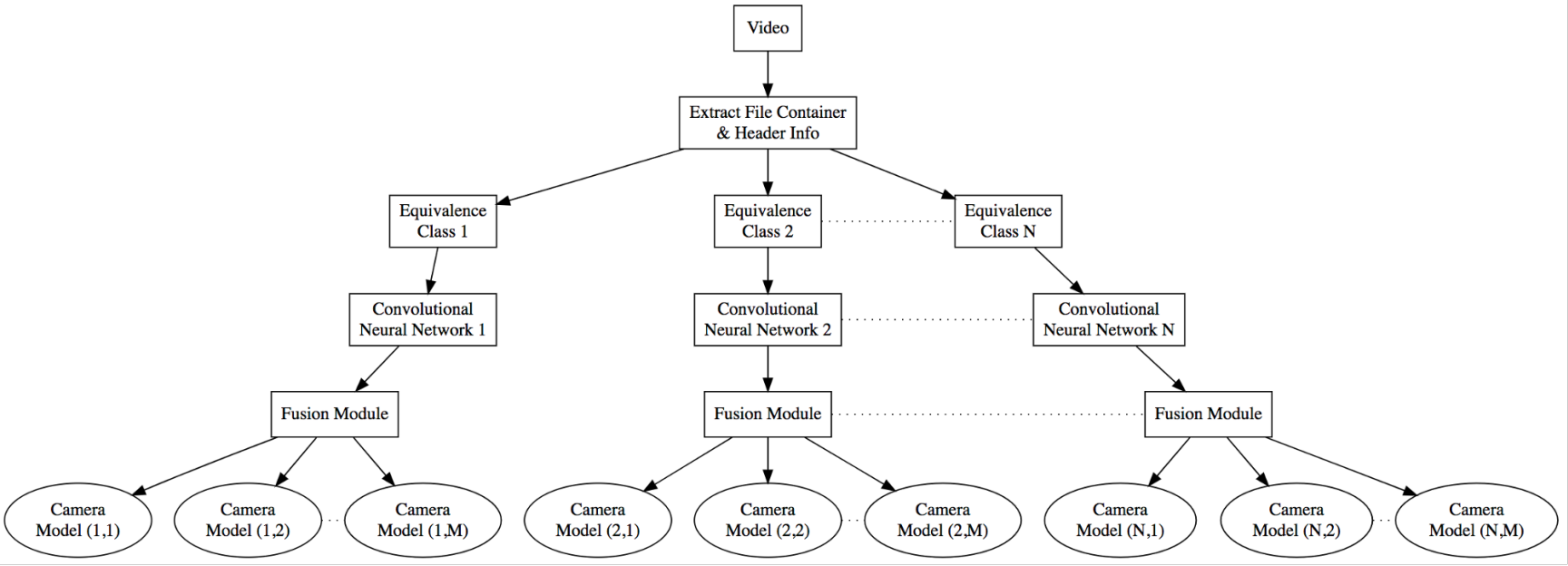


Figure 3: Flowchart providing an overview of our camera model identification framework (including hierarchical fusion of traces).

thermore, it can compare traces in two videos with completely unknown source models and let the investigator know if they originated from the same model of source camera.

This development and implementation of this system addresses Tasks 1, 2, & 4 of this project.

Algorithm Overview

This system operates by comparing the forensic traces in two input videos and outputs a score indicating whether or not these videos were captured by the same source camera model. This score takes values between 0 and 1, and can be interpreted as the likelihood that the two videos share the same source camera model (i.e. the lowest score 0 corresponds to different source camera models with high confidence and the highest score 1 corresponds to the same source camera model with high confidence). An overview of our system is shown in Figure 4.

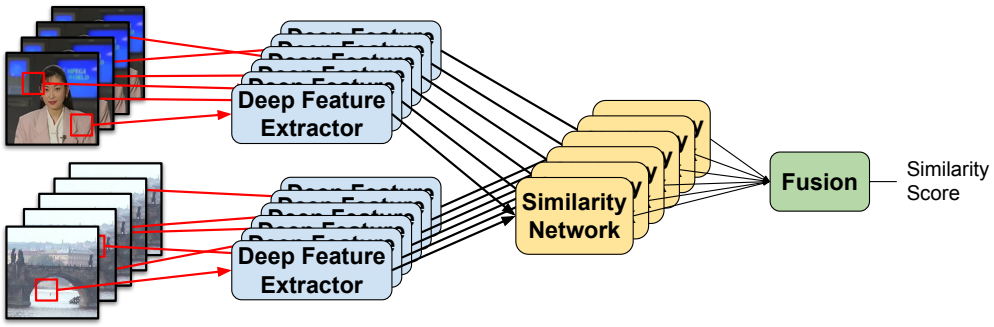


Figure 4: System diagram for open set video-level camera model verification system.

When two input videos are provided to this system, our system first chooses patch locations for forensic feature extraction. This is done by randomly choosing F I-frames from each video, then randomly selecting N patches per frame. Forensic traces are then extracted from each patch using the system described in Section 1.2.1 (this is referred to as the Deep Feature Extractor in Figure 4. This produces a total of $K = N \times F$ forensic trace vectors from each video. In practice, we have found that our system performs well when $F = 6$ frames are used and $N = 10$ patches per frame are used, producing a total of $K = 60$ forensic trace measurements per video.

Next, the forensic traces in a patch from one video are compared to the traces in a different patch from the second video. This is done using a "similarity" neural network. The combination of the forensic trace extraction CNN and the similarity network form a special neural network referred to as a Siamese network. A system diagram of our forensic siamese network is shown in Figure 5 [4]. Our siamese network has two output neurons, one which activates if the source camera model traces from both patches are similar and another which activates if they are dissimilar. The output of these two neurons are measured for a total of P randomly chosen pairings of the forensic traces in the K patches chosen from each video. In practice, we have found that our system performs well when the similarity between $P = 100$ pairings are measured.

Each of the output measurements of our forensic siamese network are stored as a two dimensional vector, then fused by averaging all of these vectors. The fused vector is then normalized by passing it through the softmax activation function. The entry of the this normalized vector corresponding to the output neuron measuring similarity is taken as our system's final output score.

A more detailed description of this system can be found in [5].

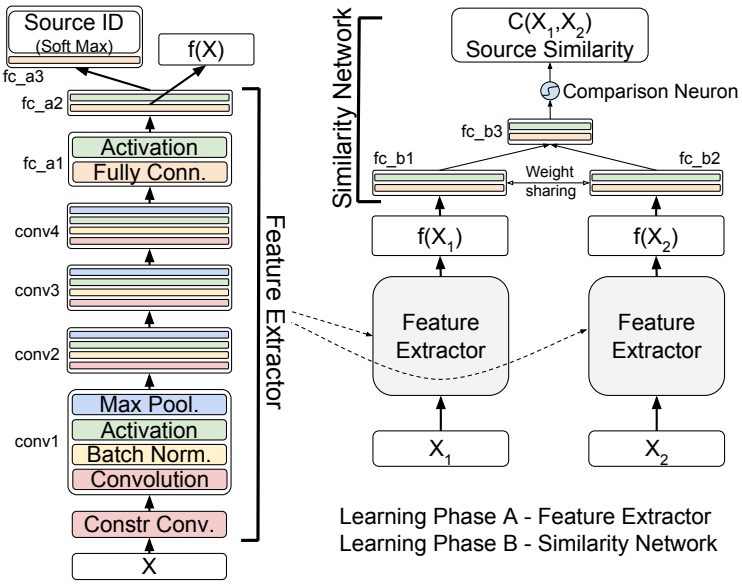


Figure 5: Architecture of our forensic Siamese network.

Testing and Validation Results

We performed multiple experiments to evaluate our open set camera model verification algorithm's performance. To do this, we formed a dataset of 33 camera models to both train and test our algorithm. This dataset corresponds to videos from all of the cameras in our dataset described in Section 1.2.8 that were capable of capturing video with a resolution of 1920×1080 pixels. Each model has at least 250 videos, with many captured by multiple devices.

First, we evaluated our system's performance determining whether or not two videos were captured by the same camera model. To do this, we grouped videos into two sets: "seen" camera models which our system encountered during training and "unseen" camera models which our system has never encountered. We measured our system's performance using 6-fold cross validation, where each fold contained a subset of training camera models, and a disjoint subset of testing camera models. Our testing set contained videos from both "seen" camera models for training and a set of videos from "unseen" camera models not encountered during training.

The average performance of our system is shown in Table 5, which contains accuracies obtained using patch-level comparisons only, as well as frame-level and video-level fusion. From this table, we can see that our system achieved 96.0% accuracy when evaluated on videos taken by camera models "seen" during training using video-level fusion. Furthermore, our system can still determine that two videos if captured by the same camera model with 87.3% accuracy even when both videos were captured by camera models never encountered during training.

Table 5: Verification rates by fusion level

	Unseen v Unseen (Open Set)	Seen v Unseen	Seen v Seen (Closed Set)
Patch	83.0%	84.2%	91.5%
Frame	85.3%	86.6%	95.3%
Video	87.3%	86.9%	96.0%

While it is common to show summary statistics on verification rates, like those shown in Ta-

ble 5, we believe it is also important to show the by-camera-model breakdown to highlight that verification rates are dependent on specific class pairings, and that different classes maybe more or less similar than others.

Next, we examined the by-camera-model breakdown of our system's performance. Fig. 6 shows the breakdown of verification rates for each pairing of camera model. Rates are shown for (a) patch-level, (b) frame-level, and (c) video-level verification. For video-level fusion, 44 out of the 55 camera model pairings achieve a very high correct verification rate of at least 95%. We note that for the pair of camera models with the lowest performance – Canon EOS SL1 and Canon Powershot SX610 – these models share a manufacturer and likely have similar processing pipelines. Overall, these results demonstrate that our open set camera model verification system can achieve strong performance.

1.2.6 Video Speed Manipulation Detection

One important manipulation that has been used in previous misinformation attempts is altering the speed of a video. For example, in 2019 a video was widely circulated on social media that had its speed manipulated to make Speaker of the House Nancy Pelosi appear disoriented.

We identified forensic traces introduced into a video file when its speed is manipulated, i.e. the video is either slowed down or sped up the video. Using these traces, we developed new algorithms to detect video speed manipulation and to estimate the rate by which a video's speed has been modified.

The development and implementation of these algorithms addressed Tasks 3 & 4 of this project.

Speed Manipulation Traces

Video speed manipulation introduces forensic traces that are visually detectable when examining the encoded frame size (EFS) sequence of a video that has been slowed down. The EFS sequence is simply the number of bytes used to encode each frame (excluding I-frames, which are not predicted).

This phenomenon can directly visible in Fig. 7, which shows the EFS sequences of an original video (a), a video that has been slowed down to 80% of its original speed (b), and a video that has been sped up to 125% of its original speed (c).

When a video is slowed down, this trace corresponds to periodic drops in the EFS to near zero. This is caused by duplicate frames introduced when creating the slowed down video, as can be seen in Figure 7(b). Speeding up video does not introduce periodic drops in the EFS sequence to values near zero, but it does introduce much more subtle periodic variations in the EFS sequence. Some frames must be skipped over when creating the sped up video sequence, creating increased prediction erro for the encoder and thus a larger encoded frame size. This introduces small but distinct periodic increases in the EFS sequence as can be seen in Figure 7(c).

Algorithm Overview

Our speed manipulation detection algorithm operates by first calculating a residual signal that isolates any potential speed manipulation traces. First, any variations in the EFS sequence that could correspond to speed manipulation traces are suppressed by median filtering the EFS sequence. Next, we subtract the trace-free sequence from the observed sequence to obtain a residual signal. If speed manipulation has occurred, this residual signal is an estimate of any speed manip-

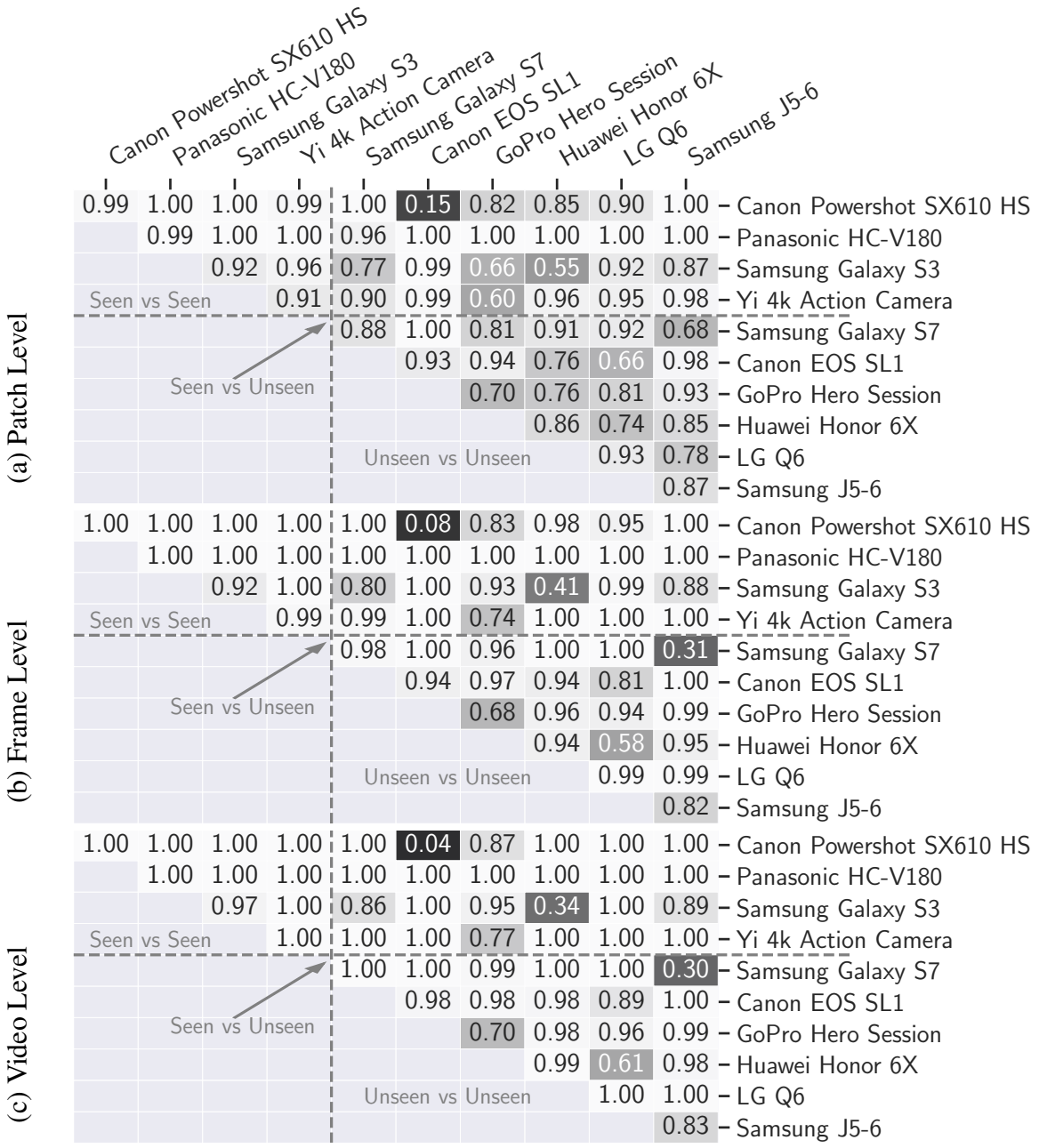
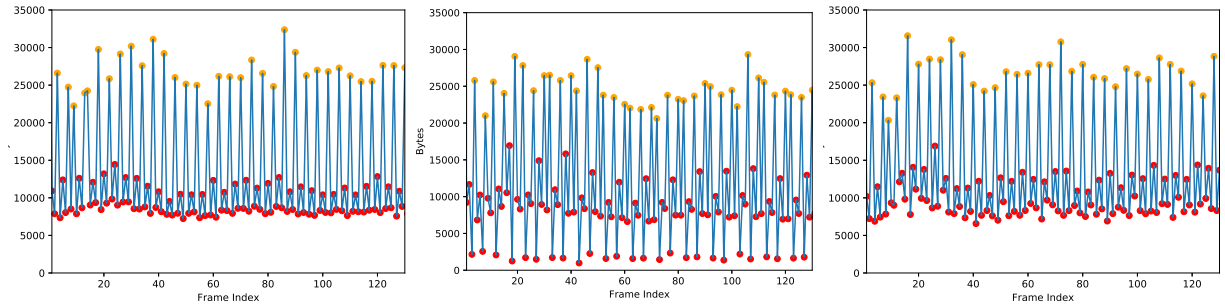


Figure 6: Verification rates between different camera models and at different fusion levels.



(a) Original video (b) Slowed to 80% speed (c) Sped up to 125% speed

Figure 7: Size (number of bytes) of encoded frames of an original video, and two manipulated copies. Red points indicate B frames and orange points indicate P frames. I frames are not shown.

ulation traces present in the video's EFS sequence. If speed manipulation has not occurred, this residual signal is simply noise.

Next, an N^{th} order AR model of the residual signal is made. The N AR model parameters are used as detection features by a support vector machine (SVM) that is trained to distinguish between speed manipulated and unaltered videos on the basis of these features. In practice, we use a 21^{st} order AR model as we have found that this results in strong detection performance while reducing the risk of overfitting.

Additionally, we developed an algorithm to estimate the amount by which a video's speed was manipulated. This is quantified by a speed manipulation parameter, which is the rate by which the video's speed has been changed. Speed manipulation parameters of less than one correspond to a slowed down video, while speed manipulation parameters correspond to a sped up video. Our speed manipulation parameter estimation algorithm operates by using the same statistical representation of speed manipulation traces that our detection algorithm uses. First, we calculate a residual signal that isolates any potential speed manipulation traces. Next, an AR model of the residual signal is made. The AR model parameters are used as features by a regression support vector machine (SVM-R) that is trained to distinguish between speed manipulated parameters of manipulated videos on the basis of these features.

A more detailed description of these algorithms can be found in [6].

Testing and Validation Results

To evaluate our algorithms, we created a dataset by selecting 414 unaltered videos from the Deepfake Detection Challenge (DFDC) dataset **[ADD CITE]**. We created a set of manipulated copies of each of these video at several different speed manipulation parameters using FFmpeg. Speed manipulation parameters between 0.6 and 1.4 selected at an interval of 0.05 were used to create manipulated version of each video. These parameters were chosen to span a range of manipulations that could reasonably fool the human eye. Our final dataset consisted of 7038 videos, corresponding to 414 unmodified videos and 6624 speed manipulated videos.

Speed Manipulation Detection

We used our dataset to train our system to identify if a video's speed has been manipulated. In this experiment, all manipulated videos belong to a single class, and all original videos belong

to another. We measured the detector's accuracy and AR model fitting time using 10-fold cross validation with a 90/10 train/test split.

We used our system to produce a Receiver Operator Characteristic (ROC) curve describing our system's performance. This ROC curve is shown below in Figure 8. This figure shows that our detector achieves a detection accuracy of 91.7% with a 9.04% probability of false alarm. From our experiment, our system acheives an area-under-the-curve (AUC) of over 0.96 (the maximum possible AUC value is 1).

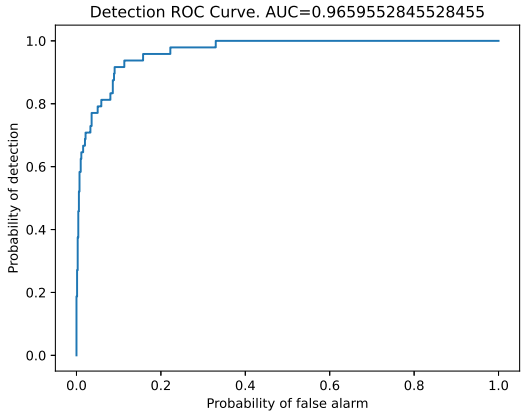


Figure 8: ROC curve obtained for our speed manipulation detector.

Additionally, we examined the performance of our detection algorithm for each inidividual speed manipualtion parameter (SMP) between 0.6 and 1.4. For each speed manipulation parameter in this range, we trained our system to discriminate between an original video and a video modified using the given speed manipulation parameter. The accuracy was measured as the average of the true positive rate and the true negative rate to counteract any class imbalances that would distort performance.

The detection accuracy achieved by our system for each speed manipulation parameter is shown in Table 6. These results show that we can reliably detect manipulated videos, achieving a maximum accuracy of 99.1% at a speed manipulation parameter of 0.7. We also see that our system performs better when detecting slowed videos (SMP $\leq$ 1), than when detecting sped-up videos, (SMP $\geq$ 1). This behavior is expected, and can be predicted when observing Figure 7. The effects of slowing a video, as shown in Figure 7(b), are much more apparent than the effects of speeding-up the same video, as in Figure 7(c).

Overall, these results shown in Table 6 and Figure 8 demonstrate that our detection system is a powerful tool for detecting manipulated videos.

Speed Manipulation Estimation

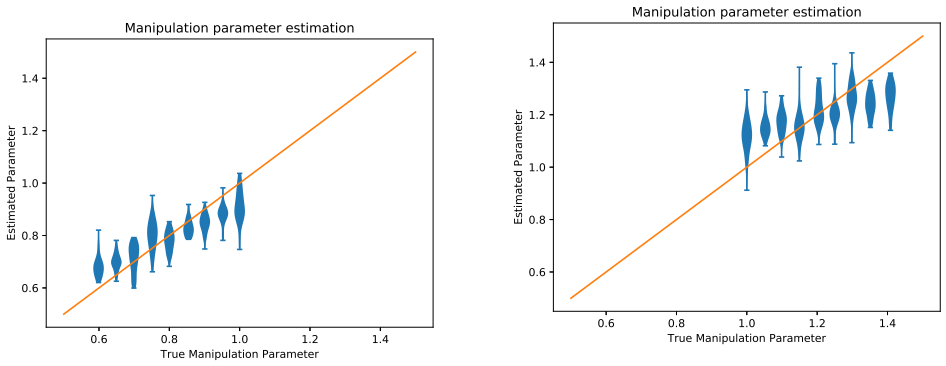
We used the same dataset of manipulated videos to evaluate the performance of our speed manipulation estimation algorithm. We note that slowed videos (SMP $<$ 1) exhibit a different fingerprint signal than sped-up videos (SMP $>$ 1) do. As a result, we investigated our system's performance by building separate SMP estimators for sped up (SMP$>$ 1) and for slowed down (SMP$<$ 1) videos.

Figure 9 shows the distribution of estimated SMP values for each video for both sped up and slowed down videos. Additionally, we measured the bias and error variance of each estimator,

Table 6: Classification accuracy of our system when considering single manipulation parameters.

Speed Manipulation Parameter (Speed Decrease)							
0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
98.8%	98.1%	99.1%	96.4%	97.0%	97.9%	97.9%	94.1%

Speed Manipulation Parameter (Speed Increase)							
1.05	1.1	1.15	1.2	1.25	1.3	1.35	1.4
83.1%	93.5%	94.6%	92.4%	90.4%	94.1%	91.6%	92.5%



(a) SMP ≤ 1

(b) SMP ≥ 1

Figure 9: Distribution of speed manipulation parameter estimations for each SMP. Ticks show maximum and minimum values.

which is listed in Table 7. These results show that our proposed system does not appear to exhibit a strong bias and achieves a low error variance. This demonstrates that our system can reasonably estimate the rate at which a video's speed has been manipulated.

Table 7: Performance of SMP estimators.

	Mean Bias	Error variance
All	-0.0025	0.02590
SMP ≥ 1	0.0011	0.01006
SMP ≤ 1	-0.0013	0.00543

1.2.7 Identifying Fake Content Using Lateral Chromatic Aberration

We developed an algorithm to identify fake content within a digital video by detecting inconsistencies in a videos' lateral chromatic aberration (LCA).

The development and implementation of this algorithm addressed Task 3 of this project.

Algorithm Overview

LCA is a minute form of color distortion that arises due to a lens's inability to focus all wavelengths of light to a common focal point on a camera's sensor. As a result, the three color channels

(red, green, and blue) in a video frame are slightly misaligned through a relative contraction or expansion about the image's optical center. This distortion, which is sometimes visible as faint purple or green fringing along the edge of an object, can be characterized as a vector describing the displacement between a point in one color channel and the same point in another color channel.

When falsifying a video, a forger often copies content from one video (i.e. the source video) then pastes it into another (i.e. the destination video). If this occurs, the LCA in the content copied from the source video is transferred over to the destination video, as is shown in Figure 10. As a result, the LCA in the falsified region deviates significantly from what would be expected in an unaltered video frame.

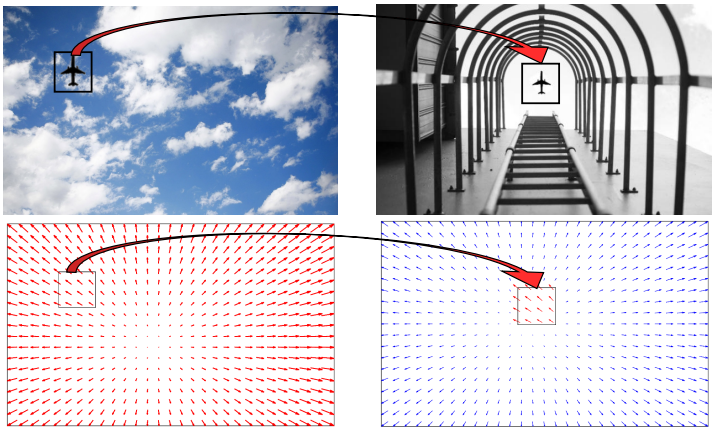


Figure 10: Example illustrating how LCA can be used to identify cut-and-paste forgeries. The vector fields below each image show simulated LCA throughout an image. The LCA in the falsified region of the frame at right is inconsistent with the LCA in the remainder of the frame.

We implemented a forensic algorithm to identify fake content within a video by detecting localized inconsistencies in the video's LCA. Our algorithm works by first efficiently estimating vectors describing the magnitude and direction of LCA at several points throughout a video frame. Next, these local estimates are used to build a global model of the expected LCA at all locations throughout a video frame. Our algorithm for estimating and modeling LCA has been extensively validated in our prior published research [7] .

Local LCA estimates are then displayed alongside the global LCA estimate at each measurement location. An investigator can then examine these LCA measurements for evidence of falsified content using our software tool. An example of this shown in Figure 11, which shows local LCA measurements in red along with modeled estimates of the expected LCA at each location for an unaltered video frame.

1.2.8 Video Database For Benchmarking Source Camera Model Identification Algorithms

To train and validate the performance of the algorithms developed under this project, we collected a large scale database of videos with known ground truth source camera models. The collection of this database addressed Task 5 of this project.

The database created under this project consists of 12,173 total videos from 46 devices, comprising 36 unique camera models. These cameras include a variety of different device categories,

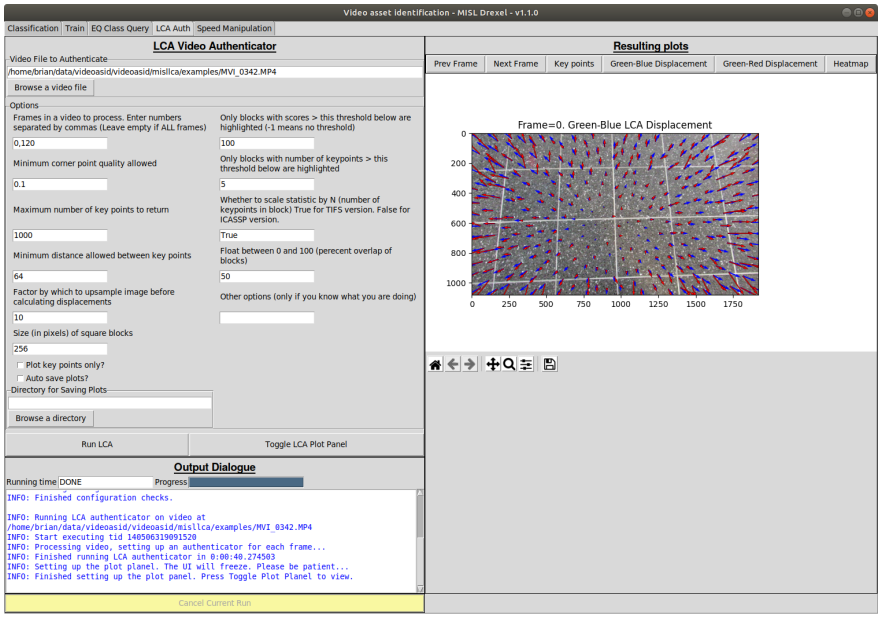


Figure 11: Example illustrating how LCA can be used to identify cut-and-paste forgeries. The vector fields below each image show simulated LCA throughout an image. This figure shows an unaltered video frame in which the LCA is consistent throughout the frame.

manufactures, and models. Specifically, our database contains videos from 19 different smartphones or tablets. We also collected videos from 10 point-and-shoot digital cameras, 3 digital camcorders, 2 single-lense reflex cameras, and 2 action cameras. Our dataset contains videos from 18 different camera manufacturers, including Apple, Asus, Canon, Fujifilm, GoPro, Google, Huawei, JVC, Kodak, LG, Motorola, Nikon, Nokia, Olympus, Panasonic, Samsung, Sony, and Yi. For nine of the camera models in the our database, videos were collected using two or more physical devices of the same make and model.

Table 8 shows the make and model of each class, as well as some properties of the videos recorded using these cameras. Videos were captured by hand by members of our research team who had physical access to each device to ensure video provenance. These videos are unaltered, in the original format directly output by the camera. Videos collected are at least 5 seconds in length and captured using the default settings of each camera. The scene content, lighting environment, indoor vs. outdoor setting, background, and other factors were varied for all videos captured by each camera model to avoid biases across the database.

Videos in the dataset were further partitioned into disjoint standardized training and testing sets. This was done to prevent overfitting for machine learning based algorithms and to standardize the evaluation procedure so that algorithms could be clearly benchmarked.

An detailed description of this dataset can be found in [8].

1.2.9 Video Authentication and Source Identification Toolkit

We developed a software suite, titled the *Video Authentication and Source Identification Toolkit*, containing software implementations of all of the algorithms developed under this project. The creation of this software addressed Task 4 of this project

Table 8: Information about the videos and source camera models in our dataset.

ID	Camera Model	Devices	Training Videos	Evaluation Videos	Average Duration (s)	Total time (s)
00	Apple iPhone 8 plus	1	223	25	5.56	1,380
01	Asus_Zenfone 3 Laser	1	234	25	6.15	1,592
02	Canon EOS SL1	2	482	25	5.58	2,829
03	Canon EOS T6i	1	202	25	5.39	1,223
04	Canon Powershot SX530 HS	1	220	25	5.98	1,466
05	Canon Powershot SX610 HS	1	191	25	5.34	1,153
06	Canon VIXIA HF R800	1	226	25	5.62	1,410
07	Fujifilm Finepix S8600	1	178	25	6.37	1,293
08	Fujifilm Finepix XP80	1	212	25	7.17	1,699
09	GoPro Hero Session	1	226	25	5.58	1,400
10	Google Pixel 1	3	753	25	6.32	4,919
11	Google Pixel 2	1	187	25	5.65	1,197
12	Huawei Honor 6X	2	476	25	5.67	2,841
13	Huawei Mate SE	1	498	25	6.36	3,325
14	JVC EverioR	1	235	25	6.46	1,681
15	Kodak Ektra	2	479	25	5.56	2,804
16	LG Q6	2	481	25	6.97	3,528
17	LG X Charge	1	234	25	5.46	1,413
18	Moto E4	2	453	25	5.35	2,558
19	Moto G5 Plus	1	439	25	5.17	2,398
20	Nikon Coolpix S33	1	196	25	6.17	1,363
21	Nikon Coolpix S3700	1	409	50	7.04	3,230
22	Nikon Coolpix S7000	1	226	25	5.42	1,361
23	Nokia 6.1	2	476	25	5.49	2,748
24	Olympus Stylus Tough TG-860	1	221	25	6.11	1,503
25	Panasonic FZ200	1	242	25	6.34	1,693
26	Panasonic HC-V180	1	240	25	6.60	1,750
27	Samsung Galaxy J7 Pro	2	408	25	6.22	2,694
28	Samsung Galaxy S3	1	230	25	5.82	1,485
29	Samsung Galaxy S5	1	257	25	5.77	1,628
30	Samsung Galaxy S7	1	206	25	5.76	1,330
31	Samsung Galaxy Tab A	1	232	25	5.46	1,402
32	Samsung J5-6	1	203	25	6.40	1,459
33	Sony Cybershot DSC-WX350	1	207	25	6.02	1,396
34	Sony Xperia L1	2	470	25	5.82	2,883
35	Yi 4k Action Camera	1	229	25	5.77	1,465
Total		46	12,173	925	5.96	71,501 s 19:51:41

The *Video Authentication and Source Identification Toolkit* is built upon a software library containing implementations of each of the algorithms discussed in Sections 1.2.1-1.2.7. Additionally, functions are provided to train each algorithm and to perform batch processing for camera model identification algorithms. Source code for this library was written using Python, TensorFlow, and C/C++ and is provided along with this toolkit. This software was developed under a Linux environment, but can be run on under Windows and MacOS environments using Docker files provided as part of this toolkit. Furthermore, the software package includes all trained machine learning algorithms and classifiers that algorithms call upon.

Users can access all functionality of this toolbox through a custom built graphical user interface (GUI). Screenshots of this GUI are shown in Figure 12. This GUI contains an output dialogue box, expandable windows to graphically display selected results, separate tabs for camera model identification, training, equivalence class queries, video speed manipulation detection, and lateral chromatic aberration based content authentication. Furthermore, it contains options for batch processing of files and for the automatic generation of reports in the JSON format detailing results of all analysis performed on a video (or videos). Alternatively, software functions in this library can be directly accessed through command line calls. This can enable technical experts to build new tools and custom scripts based upon the algorithmic capabilities provided by this library.

A 59 page user manual was created for the *Video Authentication and Source Identification Toolkit* to provide documentation and user instructions. The user manual includes directions on how to install the software package, and how to perform source camera model identification, training, and video authentication using both the GUI and a command line interface. Additionally, this documentation contains descriptions of all functions included in our software library, examples of how to call specific functions, and relevant technical information about how several algorithms operate.

Copies of this software package, including all documentation, source code, trained classifiers, and relevant executable files, were delivered to representatives from the Defense Forensics Science Center.

1.3 Opportunities for training and professional development

This project afforded several opportunities for training and professional development.

Over the course of this project, a total of seven PhD students (two of whom were women), two joint BS/MS students, and two undergraduate students (one of whom became a PhD student upon graduation) were involved in the algorithm development, software creation, and data collection performed under this project. As a result, this project enabled these students to receive advanced training in multimedia forensics, signal processing, and machine learning research from PI Stamm with additional training from Co-PI Shackleford and Co-PI Kandasamy.

During the course of this project, three PhD students (Belhassen Bayar, Chen Chen, and Owen Mayer) and one joint BS/MS (Hunter Kippen) who worked on this project defended their theses. Research performed under this project formed a part of all of these these. Additionally, one undergraduate student (Brian Hosler) who worked on this project graduated with his BS degree, then joined Drexel University as a PhD student and continued to perform doctoral research as part of this project.

Additionally, this project facilitated the professional development of both PI Stamm and the graduate students involved in this project by enabling attendance at major research conferences.

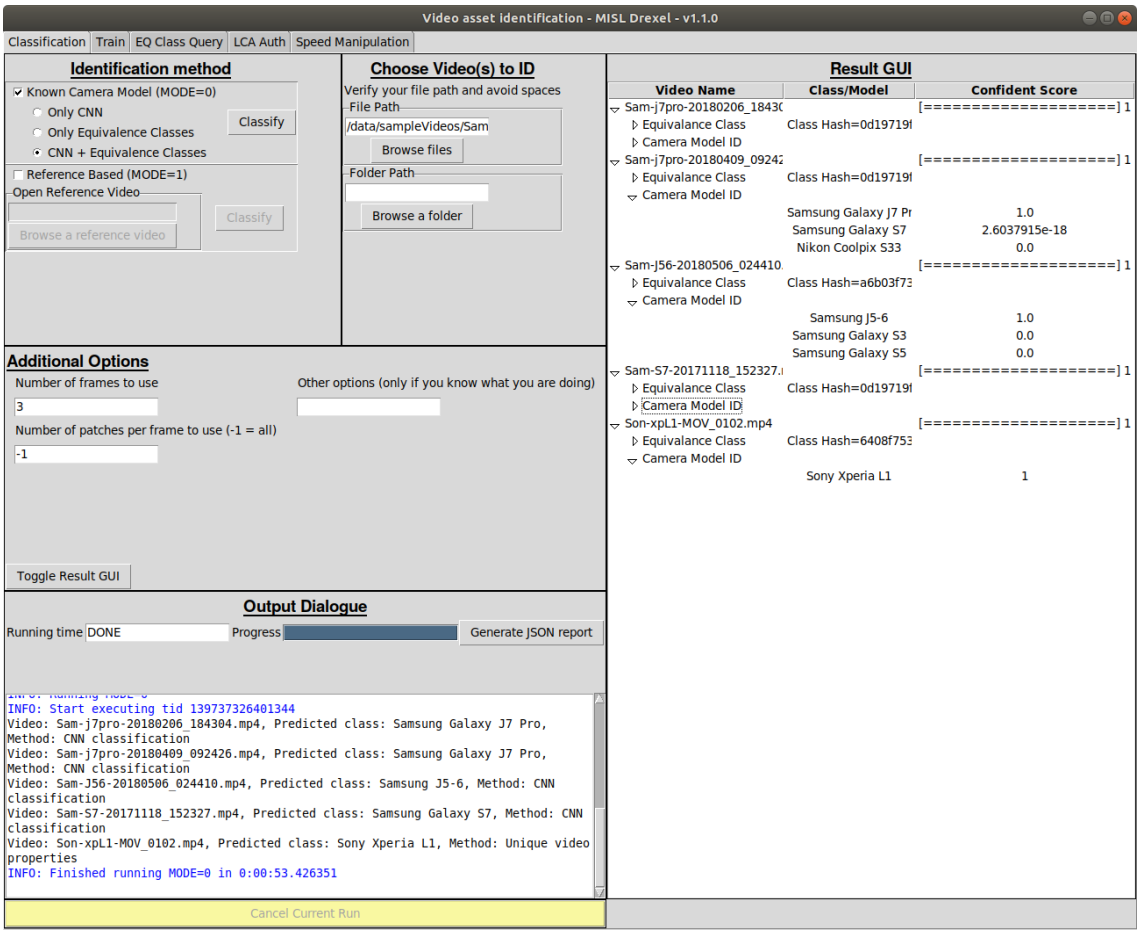


Figure 12: Screenshot of our graphical user interface when performing source camera model identification

.

Both PI Stamm and graduate students involved with this project were able to attend and present research conducted under this project at the 2019 IEEE International Conference on Acoustics, Speech, and Signal Processing, the 2020 IEEE International Conference on Acoustics, Speech, and Signal Processing, and the 2020 Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Media Forensics.

1.4 Dissemination of results to communities of interest

This project led to the publication of four papers listed in Section 2.1. Talks associated with three of these papers were given at the the 2019 IEEE International Conference on Acoustics, Speech, and Signal Processing, the 2020 IEEE International Conference on Acoustics, Speech, and Signal Processing, and the 2020 Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Media Forensics.

In addition to these publication efforts, descriptions of the algorithms developed under this project, software demonstrations, and training sessions were provided to stakeholders during performance reviews and project closeout. Furthermore, results from this project were disseminated through the following talks given by PI Stamm to other academic researchers, members of the media, and the general public:

- *Multimedia Forensics - Using AI to Identify Fake Media*, Presented at "Addressing Deepfakes in the 2020 U.S. Elections - A Conversation Among Journalists, Industry and Academia", an event run by Microsoft and the University of Washington's Center for an Informed Public, July 2020

- *Multimedia Forensics - Detecting Fake Images and Videos Using AI and Multimedia Forensics*, Presented to ECE 101, Drexel University, February 2020

- *Easy To Fool? Journalism In The Age Of Deepfakes* - South by Southwest (SXSW), with Jeremy Gilbert (The Washington Post), Paul Cheung (Knight Foundation), and Kelly Mcbride (The Poynter Institute), March 2019

- *Multimedia Forensics - Using Machine Learning to Detect Image and Video Forgeries*, Presented at Temple University, January 2019

- *Multimedia Forensics - Using Mathematics and Machine Learning to Detect Image Forgeries*, Presented at Villanova University, October 2018.

- *Multimedia Forensics - Using Mathematics and AI to Detect Multimedia Forgeries*, Presented at the Massachusetts Institute of Technology as part of the event "Preparing for the Future of Disinformation", September 2018.

- *Image Forgery Detection and Falsification Using Deep Learning*, Presented at meeting organized by WITNESS (human rights organization) & Harvard's Shorenstein Center to discuss malicious uses of Deepfakes and other AI-generated synthetic media, June 2018.

2 Products

2.1 Publications, conference papers, and presentations

The following peer-reviewed publications were produced as a result of work performed under this award:

Brian Hosler, Owen Mayer, Belhassen Bayar, Xinwei Zhao, Chen Chen, James A. Shackleford, and Matthew C. Stamm, "A video camera model identification system using deep learning and fusion." *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 8271-8275.

Status: Published
Acknowledgment of Federal Support: Yes

Brian C. Hosler, Xinwei Zhao, Owen Mayer, Chen Chen, James A. Shackleford, and Matthew C. Stamm, "The video authentication and camera identification database: A new database for video forensics." *IEEE Access*, vol. 7, Jun. 2019, pp. 76937-76948.

Status: Published
Acknowledgment of Federal Support: Yes

Owen Mayer, Brian Hosler, and Matthew C. Stamm. "Open set video camera model verification." *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 2962-2966.

Status: Published
Acknowledgment of Federal Support: Yes

Brian C. Hosler and Matthew C. Stamm. "Detecting Video Speed Manipulation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2020, pp. 670-671.

Status: Published
Acknowledgment of Federal Support: Yes

Furthermore, additional papers based on the work performed under this project are planned or are currently being prepared. Should these papers be accepted for peer reviewed publication, an acknowledgement of federal support from this project will be included.

2.2 Databases

A video database was collected in order to perform testing and validation of the camera model identification algorithms and software tools developed under this award. This database contains more than 12,000 videos from 46 physical devices representing 36 unique camera models. Videos in this database are hand collected in a diversity of real-world scenarios, are unedited, and have known and trusted provenance. A more detailed description of this dataset is provided in Section 1.2.8 as well as in [8].

Copies of this database were delivered to the Defense Forensics Science Center (DFSC). We note that this image database is signficantly larger than any existing database of videos commonly used by the multimedia forensics community. To the best of our knowledge, it is the only database explicitly built for the purpose of testing and validating video source camera model identification algorithms.

2.3 Technologies or techniques

This award led to the development of a software package titled the *Video Authentication and Source Identification Toolkit*, containing software implementations of all of the algorithms developed under this project. This software package is accompanied with a 59 page user manual providing usage, installation instructions, and code documentation. This software package, complete source code, trained classifiers, an documentation were delivered to the Defense Forensics Science Center. More details are provided in Section 1.2.9

ocumentation and user instructions. The user manual includes directions on how to install the software package, and how to perform source camera model identification, training, and video authentication using both the GUI and a command line interface. Additionally, this documentation contains descriptions of all functions included in our software library, examples of how to call specific functions, and relevant technical information about how several algorithms operate.

Copies of this software package, including all documentation, source code, trained classifiers, and relevant executable files, were delivered to representatives from the Defense Forensics Science Center.

3 Participants & Other Collaborating Organizations

3.1 Participants

- Matthew C. Stamm, Ph.D. (PI)
  - Role: Principal Investigator
  - Contribution to Project: Project management, algorithm development
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- James Shackleford, Ph.D. (Co-PI)
  - Role: Co-Principal Investigator
  - Contribution to Project: Project management, algorithm development
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- Nagarajan Kandasamy, Ph.D. (Co-PI)

- Role: Co-Principal Investigator
- Contribution to Project: Project management, algorithm development
- Collaborated With Individual in Foreign Country: No
- Traveled to Foreign Country: No

- Belhassen Bayar
  - Role: Research Assistant (Ph.D. Student)
  - Contribution to Project: Programming/algorithm implementation, algorithm development, video training data acquisition
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- Chen Chen
  - Role: Research Assistant (Ph.D. Student)
  - Contribution to Project: Programming/algorithm implementation, algorithm development, video training data acquisition
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- Shengbang Fang
  - Role: Research Assistant (Ph.D. Student)
  - Contribution to Project: Programming/algorithm implementation, video training data acquisition, algorithm development
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- Brian Hosler
  - Role: Research Assistant (Ph.D. Student, began project as Undergraduate Researcher)
  - Contribution to Project: Programming/algorithm implementation, video training data acquisition, algorithm development
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- Owen Mayer
  - Role: Research Assistant (Ph.D. Student)
  - Contribution to Project: Programming/algorithm implementation, algorithm development, video training data acquisition

- Collaborated With Individual in Foreign Country: No
- Traveled to Foreign Country: No

- Michael Spanier
  - Role: Research Assistant (Ph.D. Student)
  - Contribution to Project: Programming/algorithm implementation, video training data acquisition
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- Xinwei Zhao
  - Role: Research Assistant (Ph.D. Student)
  - Contribution to Project: Programming/algorithm implementation, algorithm development, video training data acquisition
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- Hunter Kippen
  - Role: Research Assistant (Joint BS/MS Student)
  - Contribution to Project: Video training data acquisition, algorithm development and implementation
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- Tai Nguyen
  - Role: Research Assistant (Joint BS/MS Student)
  - Contribution to Project: Programming/algorithm implementation, graphical user interface design and implementation
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

- Jacob Baron
  - Role: Undergraduate Research Assistant
  - Contribution to Project: Programming/algorithm implementation, video training data acquisition
  - Collaborated With Individual in Foreign Country: No
  - Traveled to Foreign Country: No

3.2 Other Collaborating Organizations

Nothing to report.

4 Impact

4.1 What is the impact on the development of the principal discipline of this project?

This project led to the development of several new development of several new forensic algorithms. Specifically, new forensic algorithms were developed to perform video source camera model identification, fuse forensic traces from videos, perform open set video source camera verification, and determine if a video's speed had been manipulated. This has led to the publication of four peer reviewed scientific papers. These publications are listed in Section 2.1.

We note that the algorithms developed under this project have significantly advanced efforts to forensically determine a video's source and authenticity. In particular, the algorithm discussed in Sections 1.2.1 and 1.2.2, and published in [3] currently produces the highest video source camera model identification accuracy reported in scientific literature to the best of our knowledge.

Furthermore, research conducted under this project led to the discovery of a previously unknown forensic trace left when a video's speed manipulation. Our algorithm that uses this trace to detect video speed manipulation is the first technique (and to the best of our knowledge currently the only technique) capable of detecting that a video's speed has been manipulated. This forensic trace and the associated detection algorithm are discussed in Section 1.2.6.

4.2 What is the impact on other disciplines?

Nothing to report.

4.3 What is the impact on physical, institutional, and information resources that form infrastructure?

This project led to the development of the image database described in Sections 1.2.8 and 2.2. This database will facilitate significant future research towards the development of new and improved forensic algorithms designed to determine the source, authenticity, and processing history of digital images.

4.4 What is the impact on society beyond science and technology?

The algorithms developed under this project can be used by the community at large to aid in verifying the source and authenticity of digital videos. In particular, these algorithms may be useful to news agencies that wish to verify the source and authenticity of their videos used in reporting, law enforcement agencies who wish to identify the source of videos (particularly those involved in child exploitation cases), legal experts who wish to verify evidence used in criminal

and civil proceedings, and military and defense organizations who wish to determine the origin and authenticity of signal intelligence.

Additionally, the algorithms and software developed under this project can be used to help identify and prevent the spread of misinformation. These algorithms can be used to flag and prevent the spread of videos have been manipulated or originate from an inauthentic source, by stakeholders such as news agencies, federal investigative agencies, and social media companies.

Additionally, the video speed manipulation trace discussed in Section 1.2.6 was used to determine that a video allegedly showing Speaker of the House Nancy Pelosi slurring her speech was in fact manipulated.[1] This video was part of misinformation that was circulating on social media in May 2019. PI Stamm and his team were contacted by a news agency, after which they analyzed this video and demonstrated that it had been slowed down to alter Speaker Pelosi's speech.

4.5 What dollar amount of the award's budget is being spent in foreign countries?

None of this project's funding was spent in foreign countries.

5 Budgetary Information

Amount funded to date: $648,572.00
Total amount invoiced: $577,109.32
Current available balance: $71,462.68

[1] https://www.politifact.com/factchecks/2019/may/24/politics-watchdog/viral-video-nancy-pelosi-speech-was-manipulated/

References

[1] B. Bayar and M. C. Stamm, "Constrained Convolutional Neural Networks: A New Approach Towards General Purpose Image Manipulation Detection," *IEEE Trans. on Information Forensics and Security*, Vol. 13, No. 11, pp. 2691–2706, Nov 2018.

[2] Belhassen Bayar and Matthew C Stamm, "Design principles of convolutional neural networks for multimedia forensics," *Electronic Imaging*, Vol. 2017, No. 7, pp. 77–86, 2017.

[3] B. Hosler, O. Mayer, B. Bayar, X. Zhao, C. Chen, J. A. Shackleford, and M. C. Stamm, "A Video Camera Model Identification System Using Deep Learning and Fusion," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8271–8275, May 2019.

[4] Owen Mayer and Matthew C. Stamm, "Forensic similarity for digital images," *IEEE Transactions on Information Forensics and Security*, Vol. 15, pp. 1331–1346, 2019.

[5] Owen Mayer, Brian Hosler, and Matthew C Stamm, "Open set video camera model verification," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2962–2966, May 2020.

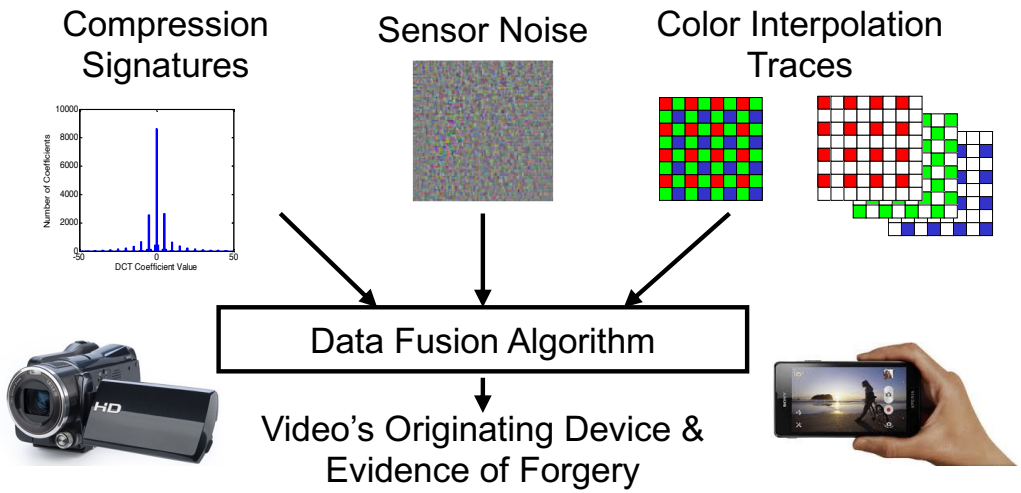
[6] Brian C Hosler and Matthew C Stamm, "Detecting Video Speed Manipulation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 670–671, June 2020.

[7] Owen Mayer and Matthew C. Stamm, "Accurate and efficient image forgery detection using lateral chromatic aberration," *IEEE Transactions on Information Forensics and Security*, Vol. 13, No. 7, pp. 1762–1777, 2018.

[8] 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.

High Performance Techniques to Identify the Source and Authenticity of Digital Videos Using Multimedia Forensics, Drexel University, ARMY22-DFSC01

Algorithms to Determine the Source Camera and Authenticity of Digital Videos



Contribution to Requirement

- Ability to determine a video's source and authenticity is critical in many scenarios (criminal investigations, signal intelligence)
- Analysts at organizations such as DC3 and NMEC require this capability
- Existing forensic techniques provide inadequate solution
- This project has developed algorithms & software tools to provide forensic analysts with these capabilities

Technology Transition – This project has produced:

- Software tool with graphical user interface
- Source code, documentation, and training data

TRL: Initial – 5 Final – 7

Project Objectives & Scope

- Provide new forensic algorithms to:
  - Identify the make and model of a video's source camera
  - Detect evidence of video manipulation and forgery
- High performance software tools based on algorithms
- Testing and validation of all software tools and algorithms

Key Personnel, Facilities

- Matthew C. Stamm (PI), Nagarajan Kandasamy (Co-PI), James A. Shackleford (Co-PI)
- Facilities: 1,200 sq. ft. lab space at Drexel University
- Contact: mstamm@drexel.edu 215-895-5894

Related Prior or Current Work

- Development of algorithms and software tools to determine a digital image's source camera – Funded through cooperative agreement with ARO and DFBA

Total Costs - $577,109 **Project Duration** - 24 Months

Milestones & Deliverables

Deliverables provided on a 6 month rolling release schedule including tech. demo, delivery of software tool snapshot, source code, documentation, and testing & validation data

Metric of Success

Testing & validation results performed on videos captured by36 different source cameras

Current Status (Final Deliverables)

- Achieved 96.60% camera model identification accuracy using deep learning approach
- Achieved 98.4% camera model accuracy using hierarchical feature fusion and deep learning
- Delivered cross-platform software implementation of all algorithms with graphical user interface (GUI), complete user documentation, and source code