# Automatic comparison of photo response non uniformity (PRNU) on Youtube

Marcel Brouwers & Rahaf Mousa[*]

UNIVERSITY OF AMSTERDAM
MASTER OF SYSTEM AND NETWORK ENGINEERING
February 12, 2017

**Abstract**

*It is possible to extract PRNU patterns from a set of videos and find a correlation to their matching source camera. Past researches have shown that this was also possible for videos uploaded to YouTube. In this research we tested if the correlation process based on the PRNU pattern still works with YouTube videos. Furthermore, we have tested which video resolutions and methods of PRNU pattern extraction are suitable for mobile phone cameras with YouTube videos. Using a test with different mobile prone cameras it is shown that this correlation is possible but depends on the type of camera. Additionally the distribution of the process of PRNU pattern extraction was tested and a method was found to limit the data transfer from YouTube.*

## I. Introduction

Cameras have become a part of our daily life, as almost every person nowadays carries at least one camera in the pocket in the form of a mobile phone. This spread of cameras and their constant usage is associated with the rapid growth of video streaming websites, such as YouTube. As each minute passes, 300 hours of videos are uploaded to YouTube, as published by Statistic Brain Research Institute in September 2016.[1] This huge amount of video uploads, in addition to the variety of content YouTube has, turned camera identification into a research target. It became especially important from the forensics point of view. Assuming there is a suspect who has taken videos or images of an unlawful act, camera identification helps in considering them as evidence even if the videos or images are deleted from the camera. Photo response non uniformity (PRNU) pattern extraction and comparison can be used to compare images and videos in order to conclude if they were possibly taken with a specific camera.

Since some of the videos might end up on video sharing websites like YouTube, it might be desirable to extract the PRNU pattern from videos on YouTube. Videos from YouTube can be downloaded and fed into software that extracts the PRNU and compares the suspected videos with a reference video made with the suspect camera. Nevertheless, when intending to process a big number of videos, the process becomes more complicated. For this reason we worked in this project on automating the process of downloading the videos in an efficient way, and feeding the downloaded material to the PRNU pattern extraction software. Afterwards the patterns from the suspected videos can be compared with the patterns extracted from the reference video.

The most important issue faced when dealing with videos downloaded from YouTube is the compression YouTube applies on videos. This

[*]Special thanks to the project supervisor dr.Zeno Geradts

compression is suspected to be affecting the PRNU making it harder to extract a good PRNU pattern.

In this project we have researched if the different video formats available on YouTube are suitable for PRNU pattern extraction. Furthermore, we have tested the PRNU extraction methods currently available in the PRNU Compare software [1] with a set of YouTube videos from mobile phone cameras. Additionally we have performed a test in which the pattern extraction is distributed over 2 machines.

This report contains six sections that consecutively discuss the theoretical background that the research is based on, the former related work, the methodology followed through out the project, the results, in addition to the conclusion and possible future work.

## II. Related Work

Using photo response non uniformity (PRNU) to identify videos from YouTube was researched by Van Houten, Wiger and Geradts [2]. In their paper, which was published in 2009, they also used the PRNU Compare software from NFI and were able to confirm that it is possible to identify the correct cameras based on videos originating from webcameras after they have been uploaded to YouTube.[2] The webcamera that was used had a native resolution of 640x480, while at the time the experiments were conducted, the highest resolution to view or download on YouTube was 480x360. That caused resizing of the uploaded videos. In addition, the videos were initially compressed by XViD or WMV before they were uploaded to YouTube.
The latest research that was conducted on camera identification on YouTube was in 2012 by Scheelen and Van der Lelie [4]. In their research, they investigated if the PRNU can be used to identify the original camera after the video has been re-encoded using the Advanced Video Codec (H.264/MPEG-4). The video resolutions they researched were 640x480 and 1280x720. The conclusion of their research was that even after a video is re-encoded, it is still possible to link some videos to their original camera.[4]

## III. Theory

In this section the most important concepts related the research are explained.

### I. Photo Response None-Uniformity (PRNU)

PRNU is a type of sensor pattern noise that is caused primarily by pixel non-uniformity (PNU). PNU is defined as different sensitivity of pixels to light caused by the inhomogenity of silicon wafers and imperfections during the sensor manufacturing process[3]. The character and origin of the PNU noise make it unlikely that even sensors coming from the same wafer would exhibit correlated PNU patterns [3], which provides a unique sensor fingerprint[5]. This fingerprint can be used in proving that a number of videos or images were taken with a specific camera, or to link images and videos taken with the same camera without having the camera in possession. This method of camera identification was proposed by Lukáš, Fridrich and Goljan in 2005.[3]

To assert a match between PRNU patterns, three steps have to be taken. The first step is the PRNU noise extraction from an image or a video's frame under investigation:

$$W = I - F(I) \tag{1}$$

where **I** is the original image, **F** the denoising operator and **F(I)** the denoised image.[6] The most well known denoising algorithms are the Wavelet filter [3], Anisotropic Diffusion (AD) filtering[7], Adaptive Spatial (AS) filtering [8] and FSTV algorithm [6].

[1]Provided to us by the Netherlands Forensic Institute

[2]In their experiments they used a wavelet filter from Lukáš *et al.* [3]

The second step is to obtain the Sensor Pattern Noise (SPN), which is the estimation of the PRNU pattern. That can be done by averaging flat-field[3] images or video's frames. There are several ways to do this, such as Basic SPN, MLE SPN and Phase SPN.[6]

The third step is to detect whether a suspect image correlates to a reference SPN. For this either the Normalized Correlation or the Peak-to-Correlation Energy (PCE) ratio can be used. Peak-to-Correlation Energy ratio was introduced by Goljan [9] as a replacement for the normalized correlation detector. According to Goljan: "Properties of PCE are especially useful when a periodic signal common to images from various cameras (like the linear pattern) enter the image noise residuals."[9] Which means that it prevents false positives when images from different cameras have a common periodic signal because they will have a lower PCE. It also lowers the PCE of true positives, but the PCE of true positives is usually so high that this difference is not significant to the end result. It should also be possible to define a universal threshold with PCE, while for normal correlation this threshold differs per camera.[10] For these reasons we have used PCE in our experiment instead of the normalized correlation.

## II. YouTube

Videos uploaded to Youtube get re-encoded in a variety of different formats. These formats are combinations of different resolutions, codecs and containers. The list of formats for a specific video can be obtained by downloading the get_video_info file for the video, which is a file in an XML based format listing video formats available for the video. Currently the get_video_info file can be obtained using the following URL[4]: `https://www.youtube.com/get_video_info?&video_id=`. The video id of the video is to be filled in after video_id=. The get_video_info file often contains a reference to the MPD DASH manifest file for streaming purposes. The MPD DASH manifest contains references to segments of the video in different resolutions. Each segment contains a short amount of playback time for the video, and the player on the client device plays the segments consecutively. This way the client side player can switch between resolutions depending on the bandwidth available and the resolution selected by the user. For the experiments described in this report, only the video formats in the get_video_info file are used since the DASH video parts would not get accepted by the PRNU Compare software. In table 1 the formats used in our experiments are listed. We chose these formats because they are available for most YouTube videos and are accepted by the PRNU Compare software.

| Itag | Resolution | Codec | Container |
|---|---|---|---|
| 17 | 176 x 144 | mp4v | 3gp |
| 18 | 640 x 360 | H.264 | mp4 |
| 22 | 1280 x 720 | H.264 | mp4 |
| 36 | 320 x 180 | mp4v | 3gp |

**Table 1:** *Itag formats used in our experiments*

## IV. Methodology

In this chapter we are going to describe the software and hardware environment used in the experiment. Furthermore, we are going to describe the approach we followed while conducting the experiments.

## I. Experimental Environment

### I.1 PRNU Compare software

The PRNU Compare software from NFI is the essential element in the experiments we conducted. The software is used to extract the PRNU pat-

[3]Images or videos which are homogeneously coloured and evenly lit, such as a white paper or a blue sky

[4]We found the URL to download these files in the source code of the youtube-dl program. It was not found in the documentation for the YouTube API so this method might not work in the future

terns from images and videos, in addition to comparing the patterns. The software consists of a Java jar file which can be used to extract the PRNU pattern from the video. The PRNU Compare software averages the PRNU extracted from a configurable number of frames and saves the extracted PRNU patterns in a pattern file. Furthermore, it keeps a list of extracted patterns in a an XML file. A special version of the PRNU Compare software that accepts CLI parameters was made available to us for the experiment in order to automate the process of PRNU extraction. The software supports 4 different filters for PRNU extraction: 2nd order (FSTV) extraction filter, 4th order extraction filter [11], wavelet Daubechies filter [3] and a wavelet Coiflet filter [3]. The required filter can be set when performing the extraction. The number of frames from the video to be taken to average the PRNU pattern can also be set using the commandline. The comparison of the PRNU patterns can currently only be done using the graphical interface of the software.

### I.2 Downloading the videos

The different video formats available for downloading of a video on YouTube are listed in the get_video_info file as explained in section II. To save developing time, the software youtube-dl is used to extract the different video formats available from the get_video_info and manifest files generated by YouTube. The different formats are indicated by Youtube using different "Itags". The output of Youtube-dl is parsed using a PHP script which selects a preferred (Itag) format. Then Youtube-dl is used again to obtain the URL of the file for the specific format. Youtube-dl can be used to download whole video files from Youtube. However, downloading the whole videos from Youtube would be a waste of bandwidth if only 200 frames of the video are needed. For this, a feature of FFmpeg comes in handy. FFmpeg can be set to download only part of the file. FFmpeg will send a TCP reset packet to the server when a big enough part of the video is received, signaling the file transfer to stop. This saves bandwidth and speeds up the processing of a batch of videos. FFmpeg can also be used to skip the first several seconds of a video in case there is a title or leader at the beginning of the video. However, the first seconds of the video that are skipped will still be downloaded.

### I.3 Search and queue management

In order to make the process of selecting the videos for PRNU extraction simple, we created a web interface to search the videos using the Youtube API. The videos can be selected and are then added to the queue (a table within a MySQL database) for processing. To start the processing of the queue, a script is set up to run using crontab. The script checks if there is anything in the queue to process. We ran a maximum of 3 instances of the PRNU Compare software at once per machine in order to efficiently use the CPU. Once added to the queue, jobs can also be removed from the queue using the web interface.

### I.4 Hardware

The cameras used to conduct the experiment were solely mobile phone cameras. In total 12 different mobile phone cameras are used in the experiment. The set consists of 5 Apple Iphone devices, a Windows Phone device and 6 Android devices of which two Huawei and 4 Samsung devices. An overview of the mobile phone cameras used in the experiments can be found in table 2.

| **Camera** | **Model** | **Recorded resolution** | **Frame rate $\approx$** |
|---|---|---|---|
| 1 | Apple Iphone 5 | 1920 x 1080 | 30 |
| 2 | Microsoft Lumia 950 | 1920 x 1080 | 25 |
| 3 | Apple Iphone 5 | 1920 x 1080 | 30 |

| Camera | Model | Recorded resolution | Frame rate ≈ |
|---|---|---|---|
| 4 | Huawei Y530 | 1280 x 720 | 30 |
| 5 | Samsung S5 | 1920 x 1080 | 30 |
| 6 | Apple Iphone 6 | 1920 x 1080 | 30 |
| 7 | Apple Iphone 6s | 1920 x 1080 | 30 |
| 8 | Apple Iphone 5s | 1920 x 1080 | 30 |
| 9 | Samsung GTI9301I | 1920 x 1080 | 30 |
| 10 | Samsung SM-G531F | 1920 x 1080 | 30 |
| 11 | Samsung Galaxy Note 2 | 1920 x 1080 | 30 |
| 12 | Huawei P8 Lite | 1920 x 1080 | 30 |

**Table 2:** *Cameras*

For extracting the PRNU patterns from the YouTube videos, we used a server with an Intel Xeon E3-1240L v5 CPU clocked at 2.10GHz and equipped with a Crucial BX200 SSD. For the distribution experiment where we let two servers work in parallel, we used an additional server equipped with the same hardware. The two servers have a 1000 Mbit link between them and a connection to the internet of adequate speed.

### I.5 Putting it all together

For letting everything work together a number of PHP scripts is made and a MySQL database is created. The MySQL database is used to keep track of what is in the queue for processing and the status of the processing. The PHP scripts are used to invoke the youtube-dl software, download the videos using FFmpeg, and extracting the PRNU patterns using the PRNU Compare software. As an extra, the PHP scripts are written such that the workload can be distributed over multiple servers to speed up the processing of the videos. The PHP scripts, set as cronjobs on the "slave server", periodically contact the master server to check if there are any videos in the queue that can be processed. This is done with an HTTP request and a video id is received back as a response if a video is in the queue for processing. The slave server will then receive a video id of a video to be processed if items are in the queue. If the slave server has successfully extracted the PRNU pattern the pattern file is sent to the master server using the HTTP POST request method, after which the extracted pattern is stored on the master server.

## V. Conducted Experiments

In our research, we have conducted three experiments which are related to each other in their results but not necessarily in their set up. In this section we will explain the approach we followed in the three experiments separately.

## I. Experiment 1: Testing video formats and extraction methods

Before starting the automation process of PRNU extraction of videos from YouTube, we were faced with a substantial question:

**To which extent is it still possible to match PRNU patterns of videos on YouTube with the originating cameras?**

The purpose of this first experiment is answering this question and find the optimal settings that can be used in the later experiments.

In order to collect the necessary data for our research we recorded videos with the mobile devices mentioned in Table 2. With each mobile phone camera we recorded three videos with a length of 40 seconds. The first video is the flat-field video of a white surface (a wall or a white paper) with a slight movement. The other two videos (which we call Natural videos) were taken in a room or outside of the build-

[5] We have done that in order to make our experiment more realistic

ing, disregarding the lighting or the repeat of scenes between the two videos.[5] In the research conducted by van Houten et al [2], they have proven that a reliable estimate is found by averaging the patterns from approximately 200 images. Depending on this conclusion, we have recorded videos of 40+ seconds that at least contain between 1000 and 1200 frames each, thus containing enough frames to average the PRNU from at least 200 frames. We have chosen this video length in order to spread the frames taken to extract the PRNU pattern.
After collecting the videos, we first uploaded all the videos, including the flat-field videos, to YouTube. When downloading the flat-field videos from YouTube and use the PRNU patterns extracted from these flat-field videos to identify the other videos, no videos could be correctly matched to their PRNU pattern extracted from the flat-field video downloaded from YouTube. We suspected that because the flat-field videos are homogeneously coloured, they are losing a lot of data in the compression YouTube applies on the uploaded videos. For this reason, we started uploading only the two natural videos to YouTube without specifying any special settings. Then we used the script we wrote to download those videos and feed them to the PRNU Compare software with the needed configurations specified in the configuration file. In the general case the configurations specified four video formats with the itags 17,18,22 and 34. 1 on page 3 We have chosen these formats because they cover low and high resolutions, in addition to the fact that those are the formats accepted by the PRNU Compare software. We also extracted the PRNU patterns using the four available methods in the PRNU Compare software.

In parallel to the aforementioned process, we have re-encoded the original flat-field videos to H.264 encoded files with resolutions corresponding to the formats from YouTube. We made sure the least possible compression was used (CRF value of 0) in order to minimize the loss in PRNU data. We did not truncate the flat-field videos but kept them at their original length of between 40 and 50 seconds. After that we fed the resulting videos to the PRNU Compare software with the four available methods of PRNU extraction. The last step was comparing the flat-field video with the whole set of natural videos that have the matching format and PRNU extraction method.

## II. Experiment 2: Identifying the correct camera from a larger set of videos

In this experiment we have tested PRNU extraction with a larger set of videos. In order to conduct this experiment, we have added 1000 videos to the queue for processing, including the Natural videos for the different mobile phone cameras we have uploaded to YouTube. The videos, other than the Natural videos from our earlier test, are videos that we selected from YouTube using the search interface that we created. The Itag formats that we processed were in the preference order: 22, 18, 36 and 17. We downloaded 30 seconds of the video and fed it to the PRNU Compare software for extracting the PRNU. For that, we have chosen the 2nd order extraction filter as it gave the most correct results in the first experiment. In addition to testing this with 30 second fragments we have repeated this experiment with 10 second fragments of the videos.
We have done this experiment in order to test if the videos can be matched to their correct cameras when part of a larger set of videos.
Secondly, we wanted to find out if we succeeded with limiting the data transferred for a large number of videos.

## III. Experiment 3: Distributing the PRNU pattern extraction over multiple machines

For this experiment we adapted the experimental environment in order to make the downloading and the pattern extraction of the videos distributed. We designated one of the servers as the master server and adapted the worker script such that it would request a video id to process from the master server via an HTTP request and then start working on the specific video. When the pattern extraction of the video is done the script will send the pattern file using an HTTP POST request method to the master server where the pattern file is then stored. Along with the pattern file, information about the pattern is sent to the master server such as the resolution and the extraction method. This information is required for building the XML file which is required by the PRNU Compare software to do the comparison of the patterns. Additionally, the error handling of the script was edited such that possible errors that occurred are also sent to the master server and show up in our interface.

After performing small tests to see if the distributed processing of the videos worked, we added the set of 1000 videos to the queue in order to be able to benchmark the distributed processing. The Itag formats that are processed are in their preferred order: 22, 18, 36 and 17. The goal of the experiment is to be able to show that it is possible to improve the speed of downloading and extracting the PRNU pattern from the YouTube videos by distributing the processing over multiple machines. We kept track of the time and the number of successfully processed videos. Additionally, we kept track of the amount of data traffic on each server. For this experiment we used two servers both equipped with an Intel Xeon E3-1240L v5 CPU clocked at 2.10GHz. We processed the set of 1000 videos 3 times with a single server and 3 times with both servers in parallel and averaged the results to be able to do a comparison. The same set of 1000 videos was used each time. For this experiment 15 seconds of the video are downloaded of which the last 10 seconds are used for the pattern extraction using 200 frames.

## VI. Results

In this section we will present the results of the three conducted experiments.

### I. Results of experiment 1: Testing video formats and extraction methods

At the end of this experiment, we had a set of results that consists of 16 comparisons for each camera.[6] In order to be able to assert that the camera is identified, both natural videos taken with a specific camera should give a PCE value that is higher than the first mismatch when compared with the matching flat-field PRNU pattern. We expect the PRNU match results to give a high difference in PCE values between the second match and the first mismatch. Analyses of the results is separated into two steps; the first step was finding the format that yields the most correct results. While the second step was finding the PRNU extraction method that yields the most correct results.

[6]For each camera we had 3 videos in four formats and for each format the PRNU was extracted using 4 different methods.

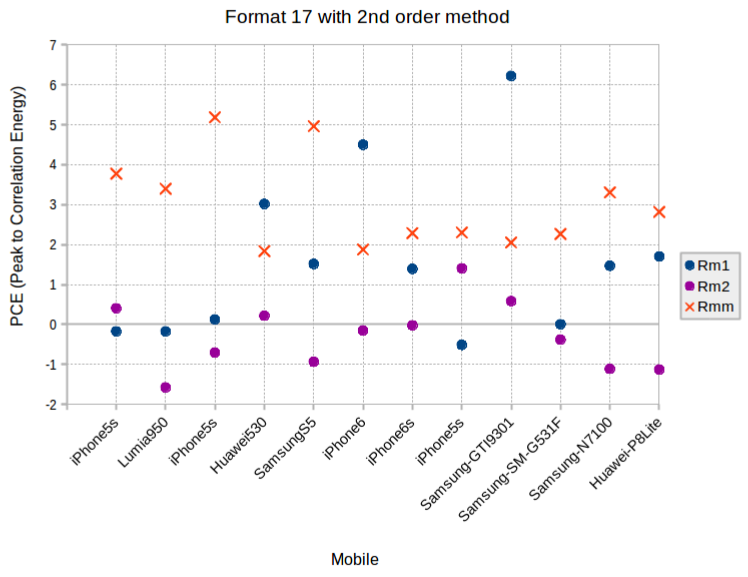

**Figure 1:** *PCE values for the 12 cameras' videos in format 17 with 2nd Order PRNU extraction method*

A sample of the format test results [7] of this experiment is presented in Figure 1. In this plot, the red X's represent the first mismatch and the other two dots represent the matches with the two natural videos. This is the plot for the test with Itag format 17 with a resolution of 176 x 144. The results presented in this plot show that that 9 of the tested cameras gave low PCE values for the matches and higher PCE values for the first mismatching pattern. Furthermore, the remaining 3 cameras gave a high PCE value for one of the matching patterns, yet the second match got a lower PCE value than the first mismatch. We also tested the matching with the Itag formats 18, 36 and 22.

A sample of the results with the Itag 22 format (1280 x 720) is presented in Figure 2 on the next page. In this plot we can see that 6 of the 12 cameras gave high PCE values for the two matching patterns which are higher than the PCE value of the first mismatching pattern. Furthermore, the remaining 6 cameras gave a PCE value for the first mismatching pattern that is either higher than one or both of the matching patterns. We have drawn this plot in the logarithmic scale because the range of the PCE values is big.[8]

Another sample of the results is presented in Figure 3 on the following page, which is a plot of the results from the Itag 22 format with the wavelet Daubechies filter. In this plot we can see that all the cameras gave a PCE value for the first mismatching pattern that is higher than one or both of the matching patterns.

[7]Based on the final results of this experiment, we have chosen to present the results of 2nd order method combined with the lowest resolution we worked with.

[8]The results contained negative values, for that reason we added the constant number 2 to the plot of the whole set in order to plot the data in the logarithmic scale.

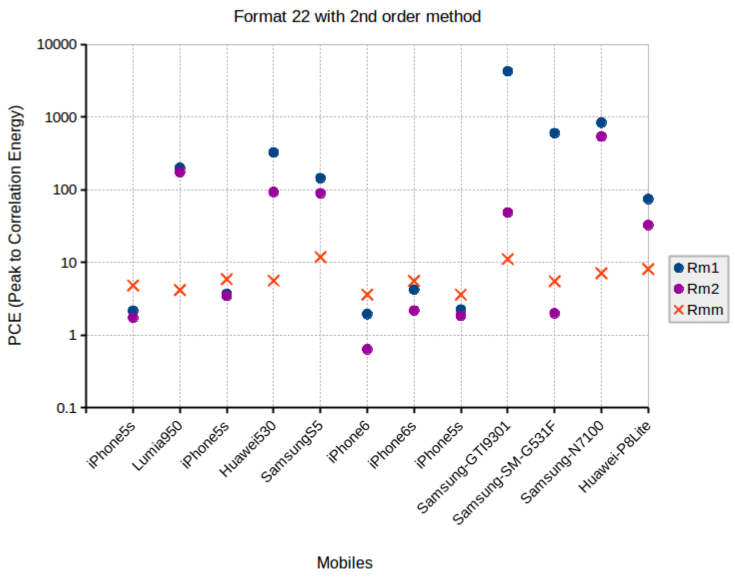

**Figure 2:** *PCE values for the 12 cameras' videos in format 22 with 2nd Order PRNU extraction method*

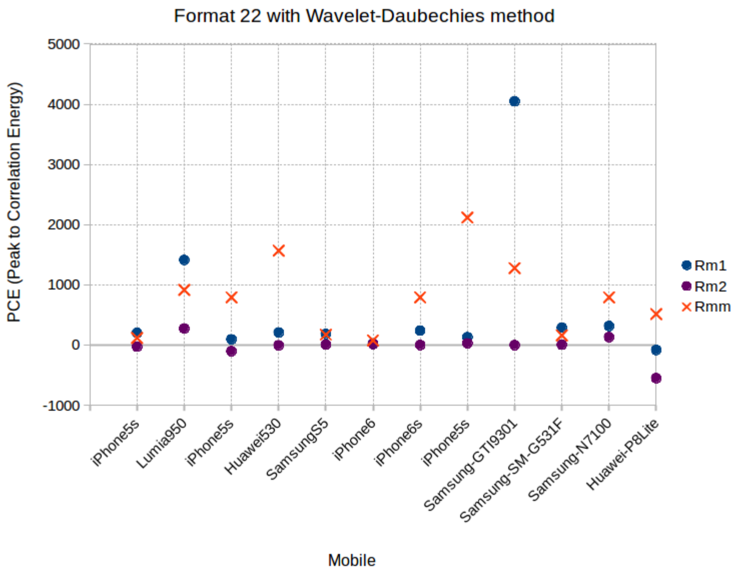

**Figure 3:** *PCE values for the 12 cameras' videos in format 22 with Wavelet Daubechies PRNU extraction method*

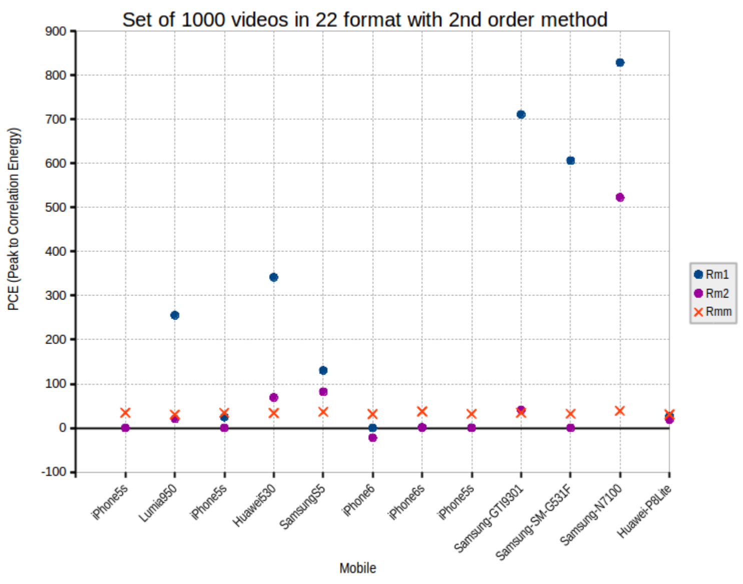

**Figure 4:** *PCE values for comparing the 962 videos with the flat-field videos*

## II. Results of experiment 2: Identifying the correct camera from a larger set of videos

The results of this experiment, in which we scale the total set of videos up to 1000 videos [9] from which we try to identify the videos from the different cameras. Out of the list of 1000 videos 962 PRNU patterns were successfully extracted [10]. The results of this experiment are presented in Figure 4. In this plot we can see that for 6 out of the 12 cameras, at least one of the two videos resulted in a PCE value higher than their mismatch, thus correctly identifying the mobile phone camera. Furthermore, for the remaining 6 cameras the PCE value for the first mismatching pattern was higher than the PCE value of the correct videos.

## III. Results of experiment 3: Distributing the PRNU pattern extraction over multiple machines

When running the experiment with the one server setup, the patterns of on average 974 videos got extracted successfully, as presented in Table 3 on the following page. For the one server setup the videos got processed in an average time of 203.2 minutes (average out of 3 tests), resulting in a rate of 288 patterns extracted per hour. In the one server setup, we can see we received about 4.16 GB (average out of 3 tests) of data from YouTube servers for the 974 videos. The two servers setup showed, as presented in Table 3 on the next page, that an average of 971 videos were downloaded and processed successfully in an average time of 97 minutes (average out of 3 tests). The rate of pattern extracting for the videos ended up around 601 videos an hour.

[9] We have collected these videos from YouTube by searching for different subjects, we have also included the video id's of the natural videos taken in the first experiment.

[10] Not all videos were always successfully processed. Videos can turn out to be live streams, deleted in the meanwhile or the formats we use for extracting are not available at the time of requesting in the get_video_info file.

| Measure (Avg. of 3) | 1 server | 2 servers |
|---|---|---|
| Successfully processed videos | 974.3 | 971 |
| Time (minutes) | 203.2 | 97 |
| Videos/hour | 288 | 601 |

**Table 3:** *1 server setup compared with the 2 server setup. Average out of 3 tests for each setup.*

## VII. Discussion

In this section we are going to discuss the results from the three experiments.

### I. Discussion of experiment 1: Testing video formats and extraction methods

The results of the first part of this experiment showed that the lower the resolution of the compared videos is, the less it is possible to correctly identify the matching camera. In the case of low resolution videos [11], presented in Figure 1 on page 8 the first mismatching pattern is giving higher PCE values than the two matching patterns in most cases.
For this reason we have excluded the low resolution video formats (Itag 36, 18 and 17) from the experiment at that point and moved on to testing the PRNU extraction methods implemented in PRNU Compare software.

In the plot shown in Figure 2 on page 9 we see that most of the cameras gave high PCE values for the matching patterns, yet low PCE values for the first mismatching pattern. That indicates that for some cameras we can correctly link videos to the originating camera. Nevertheless, the results for a number of the cameras showed that linking videos to their originating cameras is not possible (i.e. with the iPhone mobiles' cameras). We have noticed in the results that the 4th order method gave matches that are close to those given by the 2nd order method, yet the 2nd order method gave higher matches. The results of comparing PRNU patterns extracted with other methods than the 2nd order method can be found in the appendix.

While the 2nd order method, yielded high matches for a number of cameras, the two Wavelet methods [12] yielded less correct matches. That can be seen clearly in Figure 3 on page 9 where we show a plot of the Duabechies method.

### II. Discussion of experiment 2: Identifying the correct camera from a larger set of videos

Results from this experiment show that it is still possible to match the videos with their originating cameras. In the results of the first experiment, we saw that the PCE values of mismatching patterns were not much higher than 10. In this experiment, we see that if the total set of videos is bigger the chances of higher mismatch values increase. Mismatch values in this experiment could go up to values above 40.

### III. Discussion of experiment 3: Distributing the PRNU pattern extraction over multiple machines

A point of notice in the third experiment is that the average number of videos processed per hour with the two server setup is more than double the rate at which the videos get processed with the one server setup. The expectation for this experiment, was to have a rate at which the videos get processed faster than the one server setup but less than double. However, one of the servers runs the database and has to store the pattern files. In the two server setup, one of the machines uses the database from the master server to request video ids for processing and storing additional information like the resolution and extraction method. Additionally all the pattern files get sent to the master server. Thus the

[11] The itag 17 represents the resolution 176 x 144

[12] Wavelet Coiflet and Wavelet Daubechies in the way the are implemented in PRNU Compare software.

master server might have less resources available for extracting the PRNU patterns, which might explain the rate of videos processed an hour is more than doubled in the two server setup.

From certain videos the PRNU pattern could not be extracted due to various reasons; such as the video being removed from YouTube in the meanwhile, the video id belonging to a livestream or that non of the itag formats we used in the experiment are available for the specific videos. Using the two server setup, it was possible to speed up the process of downloading and extracting the PRNU patterns from the videos in our test set. For the two server test, the same set of videos used to test the 1 server configuration. Just like the 1 server setup, we ran the test of the two server setup three times and averaged the results. This is a rate which is more than twice as fast as we measured for the one server setup.

## VIII. Conclusions

Results from the experiments show that it is possible, for videos made with certain types of mobile phone cameras, to extract PRNU patterns from YouTube videos and find a correlation with their matching camera. However the success of this is dependent on the type of mobile phone camera. The Itag 22 format (1280 x 720), the highest resolution we tested, combined with the 2nd order (FSTV) filter as currently implemented in the PRNU Compare software, gives the most correct correlations at this moment in time. It can also be concluded that the process of downloading videos from YouTube and extracting the PRNU pattern can be automated. In the process, it has proven possible to limit the amount of data transferred by downloading a segment of the video. Additionally it is demonstrated that it is possible to distribute the process of PRNU pattern extraction in order to speed up the process of PRNU pattern extraction for larger sets of videos.

## IX. Future Work

Within this experiment the pattern extraction was distributed over multiple servers. However, the matching of the patterns against the flat-field patterns takes place on a single machine. For future work it might be interesting to research the distributing of this process. Furthermore it would be interesting to perform the tests with more different devices and perhaps a multiple for each device. Finally the indexing and fast search of the PRNU Patterns might be a topic of interest for future work.

## X. Acknowledgements

We would like to thank the Netherlands Forensic Institute for providing us with the PRNU Compare software for this experiment. Additionally we would like to thank Zeno Geradts for supervising the project.

## References

[1] Statistic Brain Research Institute. Youtube company statistics. `http://www.statisticbrain.com/youtube-statistics/`, 2016. URL `http://www.statisticbrain.com/youtube-statistics/`.

[2] Wiger Van Houten and Zeno Geradts. Using sensor noise to identify low resolution compressed videos from youtube. In *International Workshop on Computational Forensics*, pages 104–115. Springer, 2009.

[3] Jan Lukas, Jessica Fridrich, and Miroslav Goljan. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security*, 1(2):205–214, 2006.

[4] Yannick Scheelen and Jop van der Lelie. Camera identification on youtube.

[5] G Chierchia, S Parrilli, G Poggi, L Verdoliva, and C Sansone. Prnu-based detection

of small-size image forgeries. In *Digital Signal Processing (DSP), 2011 17th International Conference on*, pages 1–6. IEEE, 2011.

[6] Floris Gisolf, Anwar Malgoezar, Teun Baar, and Zeno Geradts. Improving source camera identification using a simplified total variation based noise removal algorithm. *Digital Investigation*, 10(3):207–214, 2013.

[7] Wiger van Houten and Zeno Geradts. Using anisotropic diffusion for efficient extraction of sensor noise in camera identification. *Journal of forensic sciences*, 57(2):521–527, 2012.

[8] Alan J Cooper. Improved photo response non-uniformity (prnu) based source camera identification. *Forensic science international*, 226(1):132–141, 2013.

[9] Miroslav Goljan. Digital camera identification from images–estimating false acceptance probability. In *International Workshop on Digital Watermarking*, pages 454–468. Springer, 2008.

[10] Floris Gisolf. Student id: 5600464 36 ec 6 februari 2012–6 july 2012 msc in forensic science. 2012.

[11] Teun Baar, Wiger van Houten, and Zeno J. M. H. Geradts. Camera identification by grouping images from database, based on shared noise patterns. *CoRR*, abs/1207.2641, 2012. URL `http://arxiv.org/abs/1207.2641`.

## A. Appendix

| Mobile phone | $\rho m1$ | $\rho m2$ | $\rho mm$ |
|---|---|---|---|
| 1. Apple Iphone 5s | -0.177129 | 0.401102 | 3.773802 |
| 2. Microsoft Lumia 950 | -0.175443 | -1.579272 | 3.395653 |
| 3. Apple Iphone 5s | 0.122929 | -0.707257 | 5.183939 |
| 4. Huawei Y530 | 3.017369 | 0.215268 | 1.836242 |
| 5. Samsung S5 | 1.513014 | -0.940633 | 4.961946 |
| 6. Apple Iphone 6 | 4.500314 | -0.152984 | 1.870462 |
| 7. Apple Iphone 6s | 1.389558 | -0.026348 | 2.286574 |
| 8. Apple Iphone 5s | -0.514208 | 1.402262 | 2.29726 |
| 9. Samsung GTI9301 | 6.225871 | 0.581295 | 2.04757 |
| 10. Samsung SM-G531F | 0.000509 | -0.382002 | 2.263757 |
| 11. Samsung Note 2 | 1.468258 | -1.114662 | 3.302357 |
| 12. Huawei P8 Lite | 1.697553 | -1.13193 | 2.815413 |

**Table 4:** *Itag 17 format with 2nd order (FSTV) filter*

| Mobile phone | $\rho m1$ | $\rho m2$ | $\rho mm$ |
|---|---|---|---|
| 1. Apple Iphone 5s | 3.34338 | 0.2548 | 8.43965 |
| 2. Microsoft Lumia 950 | 9.903549 | 2.95854 | 2.194641 |
| 3. Apple Iphone 5s | 1.278622 | -4.504052 | 8.296134 |
| 4. Huawei Y530 | 4.488281 | 1.830243 | 4.488223 |
| 5. Samsung S5 | 7.975279 | 0.106543 | 4.569173 |
| 6. Apple Iphone 6 | 8.795529 | 0.416817 | 6.039873 |
| 7. Apple Iphone 6s | 0.431319 | 0.018963 | 4.215653 |
| 8. Apple Iphone 5s | 0.000995 | -0.218826 | 2.9229921 |
| 9. Samsung GTI9301 | 101.031067 | -0.225542 | 6.620474 |
| 10. Samsung SM-G531F | 16.907707 | 0.180126 | 5.267784 |
| 11. Samsung Note 2 | 21.78717 | 0.44343 | 2.17736 |
| 12. Huawei P8 Lite | 7.109772 | 6.211234 | 2.95071 |

**Table 5:** *Itag 36 format with 2nd order (FSTV) filter*

| Mobile phone | $\rho m1$ | $\rho m2$ | $\rho mm$ |
|---|---|---|---|
| 1. Apple Iphone 5s | 6.202249 | 0.006859 | 1.493879 |
| 2. Microsoft Lumia 950 | 8.6815 | 4.632438 | 6.879198 |
| 3. Apple Iphone 5s | 1.510593 | -1.72916 | 1.281656 |
| 4. Huawei Y530 | 7.338791 | 0.467483 | 1.930481 |
| 5. Samsung S5 | 10.599456 | 7.497791 | 4.655661 |
| 6. Apple Iphone 6 | 1.831168 | -0.173506 | 3.182783 |
| 7. Apple Iphone 6s | 0.000041 | -0.025312 | 6.221945 |
| 8. Apple Iphone 5s | 4.277181 | -0.472199 | 2.407648 |
| 9. Samsung GTI9301 | 110.452377 | 3.19986 | 3.6513 |
| 10. Samsung SM-G531F | 42.35508 | 1.274687 | 5.795496 |
| 11. Samsung Note 2 | 61.931297 | 12.701159 | 4.719409 |
| 12. Huawei P8 Lite | 15.888545 | 1.246686 | 3.297715 |

**Table 6:** *Itag 18 format with 2nd order (FSTV) filter*

| Mobile phone | $\rho m1$ | $\rho m2$ | $\rho mm$ |
|---|---|---|---|
| 1. Apple Iphone 5s | 0.145145 | -0.275735 | 2.792469 |
| 2. Microsoft Lumia 950 | 197.514832 | 171.500107 | 2.125504 |
| 3. Apple Iphone 5s | 1.654127 | 1.438749 | 3.83806 |
| 4. Huawei Y530 | 320.819855 | 90.433739 | 3.550675 |
| 5. Samsung S5 | 141.668884 | 86.725296 | 9.756674 |
| 6. Apple Iphone 6 | -0.072351 | -1.368708 | 1.576358 |
| 7. Apple Iphone 6s | 2.199716 | 0.157972 | 3.519242 |
| 8. Apple Iphone 5s | 0.230641 | -0.166466 | 1.576358 |
| 9. Samsung GTI9301 | 4237.684082 | 46.260273 | 9.040846 |
| 10. Samsung SM-G531F | 596.160645 | -0.021758 | 3.468436 |
| 11. Samsung Note 2 | 833.377563 | 536.378357 | 5.052656 |
| 12. Huawei P8 Lite | 72.100372 | 30.47897 | 6.04894 |

**Table 7:** *Itag 22 format with 2nd order filter*

| Mobile phone | $\rho m1$ | $\rho m2$ | $\rho mm$ |
|---|---|---|---|
| 1. Apple Iphone 5s | -0.007655 | -0.177742 | 4.060404 |
| 2. Microsoft Lumia 950 | 131.665619 | 127.012032 | 2.402721 |
| 3. Apple Iphone 5s | 7.009462 | 0.197778 | 1.984898 |
| 4. Huawei Y530 | 311.620087 | 87.657043 | 2.970427 |
| 5. Samsung S5 | 82.196548 | 73.240982 | 4.761385 |
| 6. Apple Iphone 6 | 0.211762 | 0.004098 | 1.423713 |
| 7. Apple Iphone 6s | 1.045292 | 0.018497 | 2.237353 |
| 8. Apple Iphone 5s | 0.273609 | -0.179469 | 1.996719 |
| 9. Samsung GTI9301 | 1425.912354 | 40.242451 | 9.248312 |
| 10. Samsung SM-G531F | 319.826385 | -0.081802 | 4.794073 |
| 11. Samsung Note 2 | 259.504517 | 183.552246 | 2.514012 |
| 12. Huawei P8 Lite | 26.853085 | 5.053776 | 4.097392 |

**Table 8:** *Itag 22 format with 4th order filter*

| Mobile phone | $\rho m1$ | $\rho m2$ | $\rho mm$ |
|---|---|---|---|
| 1. Apple Iphone 5s | 71.464409 | -65.574638 | 323.76297 |
| 2. Microsoft Lumia 950 | 1883.948486 | 739.239197 | 1789.522827 |
| 3. Apple Iphone 5s | 38.975536 | -577.012756 | 3482.82251 |
| 4. Huawei Y530 | 521.487183 | -108.956558 | 3230.001465 |
| 5. Samsung S5 | 402.668549 | -5.669655 | 495.675171 |
| 6. Apple Iphone 6 | 9.98511 | -1.136031 | 149.016724 |
| 7. Apple Iphone 6s | 336.165955 | -48.495636 | 633.075867 |
| 8. Apple Iphone 5s | 262.377716 | 1.13964 | 5704.317871 |
| 9. Samsung GTI9301 | 5258.844727 | -25.617983 | 2045.573835 |
| 10. Samsung SM-G531F | 129.556442 | 14.474392 | 166.177322 |
| 11. Samsung Note 2 | 476.210541 | 141.615631 | 2704.412598 |
| 12. Huawei P8 Lite | 22.991848 | -20.66033 | 691.578369 |

**Table 9:** *Itag 22 format with Coiflet filter*

| Mobile phone | $\rho m1$ | $\rho m2$ | $\rho mm$ |
|---|---|---|---|
| 1. Apple Iphone 5s | 199.967514 | -28.097988 | 114.643532 |
| 2. Microsoft Lumia 950 | 1412.478394 | 272.581665 | 913.896606 |
| 3. Apple Iphone 5s | 93.188461 | -104.270889 | 789.3797 |
| 4. Huawei Y530 | 206.852509 | -5.548405 | 1565.675293 |
| 5. Samsung S5 | 182.858932 | 4.945631 | 167.52092 |
| 6. Apple Iphone 6 | 24.318363 | 15.528453 | 74.66848 |
| 7. Apple Iphone 6s | 237.758667 | -1.705553 | 790.57373 |
| 8. Apple Iphone 5s | 128.020248 | 26.211637 | 2114.861572 |
| 9. Samsung GTI9301 | 4049.134521 | -2.992535 | 1276.855103 |
| 10. Samsung SM-G531F | 286.575348 | 4.034087 | 157.957336 |
| 11. Samsung Note 2 | 313.225067 | 129.451965 | 789.688782 |
| 12. Huawei P8 Lite | -84.045197 | -552.218628 | 512.134521 |

**Table 10:** *Itag 22 format with Daubechies filter*

| Mobile phone | $\rho m1$ | $\rho m2$ | $\rho mm$ |
|---|---|---|---|
| 1. Apple Iphone 5s | 0.120625384 | -0.19932938 | 34.676517 |
| 2. Microsoft Lumia 950 | 255.418015 | 20.539165 | 30.340464 |
| 3. Apple Iphone 5s | 24.637943 | 0.06275389 | 33.953735 |
| 4. Huawei Y530 | 341.71475 | 69.018105 | 33.69216 |
| 5. Samsung S5 | 130.32199 | 82.029434 | 36.10736 |
| 6. Apple Iphone 6 | 0.14599928 | -22.364832 | 31.247087 |
| 7. Apple Iphone 6S | 1.1176726 | 0.92868745 | 37.313114 |
| 8. Apple Iphone 5S | 0.58487904 | -0.06048749 | 31.47116 |
| 9. Samsung GTI9301 | 711.0564 | 40.74926 | 34.070354 |
| 10. Samsung SM-G531F | 606.14417 | 0.08128629 | 31.981623 |
| 11. Samsung Note 2 | 828.44196 | 522.95123 | 38.41599 |
| 12. Huawei P8 Lite | 25.972715 | 18.742098 | 30.639076 |

**Table 11:** *Results of 1000 videos with 2nd order filter, itag 22, 30 seconds analyzed per downloaded video*

| Mobile phone | $\rho m1$ | $\rho m2$ | $\rho mm$ |
|---|---|---|---|
| 1. Apple Iphone 5s | 20.49686 | 0.48957378 | 35.357864 |
| 2. Microsoft Lumia 950 | 99.13967 | -28.724852 | 30.192146 |
| 3. Apple Iphone 5s | -0.19546966 | 0.25030282 | 30.065182 |
| 4. Huawei Y530 | 23.832335 | -25.778994 | 38.79837 |
| 5. Samsung S5 | 19.57411 | -31.101841 | 32.05532 |
| 6. Apple Iphone 6 | -22.364832 | 0.046843186 | 33.034157 |
| 7. Apple Iphone 6s | 1.1813881 | 0.03744992 | 34.886024 |
| 8. Apple Iphone 5s | 1.8863258 | 0.7486062 | 30.697187 |
| 9. Samsung GTI9301 | 1408.4209 | 19.690107 | 45.514305 |
| 10. Samsung SM-G531F | 252.10266 | 18.53082 | 32.14834 |
| 11. Samsung Note 2 | 465.95117 | 23.588686 | 36.34564 |
| 12. Huawei P8 Lite | 48.888634 | 10.357987 | 30.504707 |

**Table 12:** *Results of 1000 videos with 2nd order filter, itag 22, 10 seconds analyzed per downloaded video*

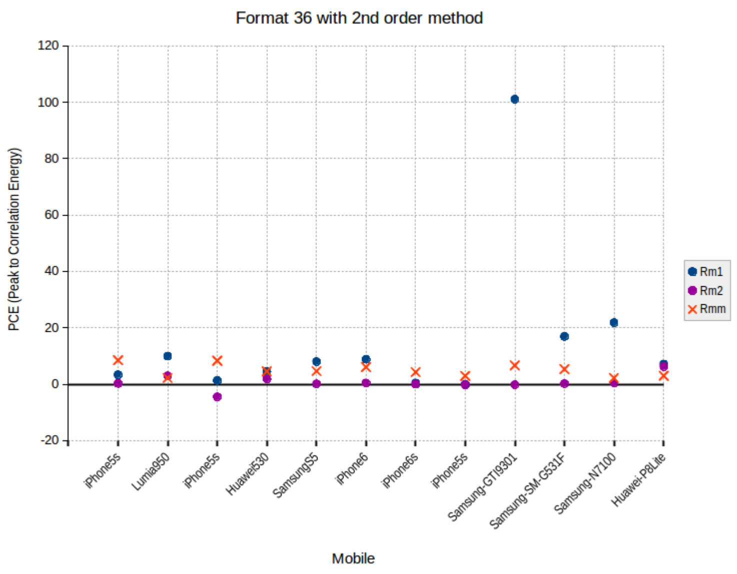

**Figure 5:** *PCE values for the 12 cameras' videos in format 18 with 2nd Order PRNU extraction method*

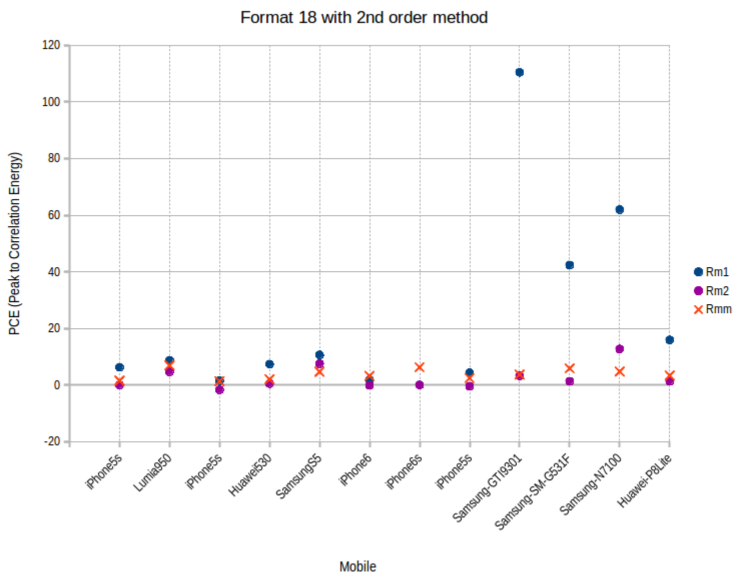

**Figure 6:** *PCE values for the 12 cameras' videos in format 36 with 2nd Order PRNU extraction method*

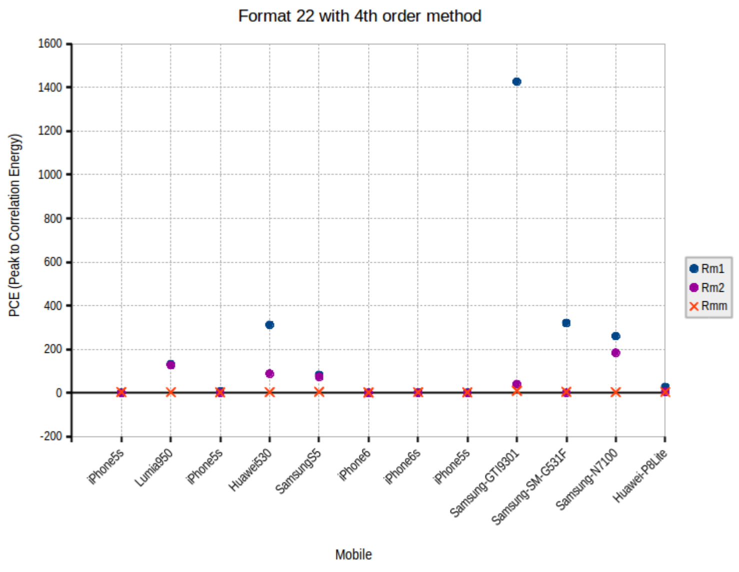

**Figure 7:** *PCE values for the 12 cameras' videos in format 22 with 4nd Order PRNU extraction method*

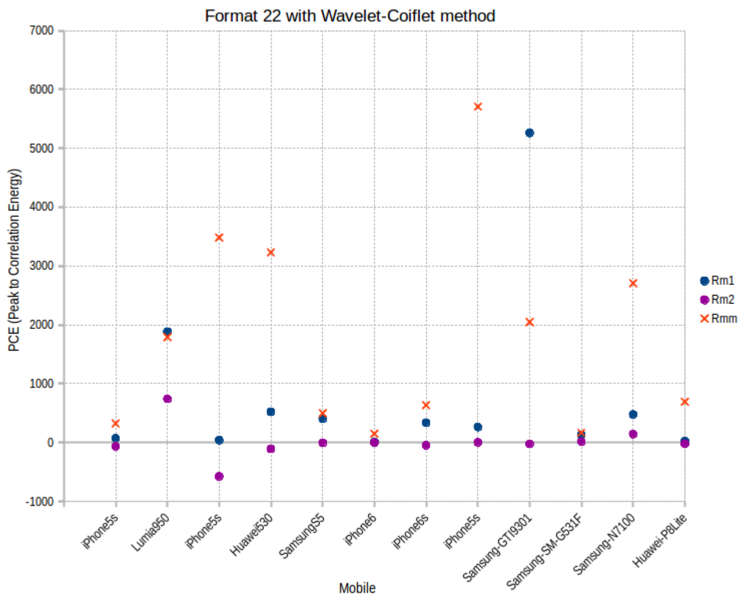

**Figure 8:** *PCE values for the 12 cameras' videos in format 22 with Coiflet PRNU extraction method*