# Using sensor noise to identify low resolution compressed videos from YouTube

Wiger van Houten and Zeno Geradts*

## Abstract

**The Photo Response Non-Uniformity acts as a digital fingerprint that can be used to identify image sensors. This characteristic has been used in previous research to identify scanners, digital photo cameras and digital video cameras. In this paper we use a wavelet filter from Lukáš *et al* [1] to extract the PRNU patterns from multiply compressed low resolution video files originating from webcameras after they have been uploaded to YouTube. The video files were recorded with various resolutions, and the resulting video files were encoded with different codecs. Depending on video characteristics (e.g. codec quality settings, recording resolution), it is possible to correctly identify cameras based on these videos.**

**Index terms** - Photo Response Non Uniformity, Video Camera Identification, Pattern Noise, Digital Forensics, YouTube, Low resolution, MSN Messenger, Windows Live Messenger

## 1 Introduction

Image sensors are being integrated in a wide range of electronics, such as notebooks and mobile phones, apart from the 'traditional' electronics as video- or photo-cameras. Forensically, the scenes and events that may be recorded with these devices may be very interesting. When, however, these videos or photographs are used in a legal context, an interesting question to be answered remains: who recorded or photographed the depicted scene? To answer this question, it may be of great help to establish which specific camera was used to record the scene. Although modern cameras often write metadata (e.g. EXchangeable Image Format (EXIF) or eXtensible Metadata Platform (XMP)) to the file containing tags as the date, time and serial number of the originating camera, there is currently no standard framework for this kind of information for video files. Also, the metadata can be removed or manipulated easily, leaving the initial quest for a reliable and more robust identification method.

*Wiger van Houten (MSc) and Zeno Geradts (PhD) are at the Digital Technology & Biometrics department of the Netherlands Forensic Institute (e-mail: w.van.houten@nfi.minjus.nl ; z.geradts@nfi.minjus.nl). This work was partially supported by the European Union sponsored FIDIS Network of Excellence.

Traditionally, defective pixels on the image sensor could be used to establish the image origin. The location of these defective pixels act as a fingerprint, as these defective pixels are present in all images the sensor produces [2]. A problem with this method occurs when no defective pixels are present, or when the image acquisition device internally corrects these defective pixels during post-processing. However, in the following years this method has been refined. Instead of looking at the location of the defective pixels, we now look at the individual pixels that may report slightly lower or higher values compared to their neighbours, when these pixels are illuminated uniformly. These small deviations form a device signature, and it is this pattern that is used for device identification. This differing sensitivity of individual pixels to light is called the Photo Response Non-Uniformity. These deviations are present in both CCD image sensors and CMOS sensors (Active Pixel Sensors, APS), which, as was mentioned above, are present in a wide range of electronic devices.

These seemingly invisible patterns originate from the sensor pattern noise, and can be used to characterise each image sensor. The origins of this pattern noise suggest that each sensor has its own unique pattern (see §2). By extracting and comparing these patterns, device identification is possible in the same way as fingerprint identification. Namely, the sensor pattern noise that is extracted from a questioned image can be compared with the reference patterns from a database of cameras, and may include the camera of a suspect. When two patterns show a high degree of similarity, it is an indication that both patterns have the same origin. Hence we conjecture that it is advantageous to build a database of patterns obtained e.g. from videos depicting child pornography; in this way the origin of different videos can be linked together. Ultimately this may aid law enforcement agencies in the prosecution of suspects.

The PRNU has been successfully used in digital photo-camera identification [3, 4, 5], scanner identification [6, 7] and also in camcorder identification [8].

Obviously, filters are needed to extract these patterns from the image. A wide range of filters is available, differing in performance and computational complexity. These filters all work by subtracting a denoised image from the original (see section 3.1). We use the wavelet filter from [1], which is based on the work in [9].

This paper is organised as follows. In section 2 the

origins of the PRNU are explained. In section 3 the algorithm from [1] used to extract the PRNU pattern from images and videos is explained. In section 4 we apply the method to low resolution videos from webcams that are subsequently uploaded to YouTube. These videos were initially compressed by XViD or WMV before they were uploaded. Finally, in sections 5 and 6 we discuss some possible future work, and conclude this paper.

## 2 Sensor noise sources

During the recording of the scene on the CMOS APS or CCD image sensor, various noise sources degrade the image. Before the light hits the silicon in the sensor, it has travelled through various components, such as (the filter of) the lens and the colour filter array[1]. Ultimately, we are interested in other characteristics, namely the 'noise' sources that originate from the sensor[2]. A distinction can be made between temporal and spatial noise. For a comprehensive overview of noise sources in CCD and CMOS digital (video) cameras, see [12, 13] and the references therein.

The main contribution to the temporal noise comes from the (photonic) shot noise and in lesser extent to the (thermal) dark current shot noise. The photonic shot noise is inherent to the nature of light, and is essentially a quantum mechanical effect due to the discrete electromagnetic field energy [14]. Due to the random arrival of photons at the image sensor, this process can be described by a Poisson distribution. This means that the variance in the amount of photons that arrive at the detector equals the mean number of photons. Hence, this temporal noise source is only significant (and observable) at very low intensity levels. These variable amounts of photons that arrive at the detector naturally generate a variable amount of electrons. The dark current shot noise follows a similar distribution, and originates from the thermal generation of charge carriers in the silicon substrate of the image sensor. As the camera is not able to differentiate the signal charge from the spurious electrons generated, these unwanted electrons are added to the output and represent a noise source. Another temporal noise source is the flicker noise, in which charges are trapped in surface states and subsequently released after some time in the charge to voltage amplifier. In CMOS active pixel sensors additional sources are present due the various transistors integrated on each pixel [15, 16]. As this temporal noise is a purely statistical phenomenon, averaging multiple frames will reduce the amount of temporal noise.

Some of the variations due to dark current are somewhat systematic: the spatial pattern of these variations remains constant. In other words, when the sensor is *not* illuminated certain systematic variations occur in the pixel values reported. Crystal defects, impurities and dislocations present in the silicon lattice may contribute to the size of the fixed pattern noise, as well as the detector size, non-uniform potential wells and varying oxide thickness. In CMOS image sensors additional sources are present, and can be thought of as composed of a column component (shared between all pixels in a certain column) and an individual pixel component [17]. For instance, due to a variable offset in the reset transistor used to reset the photodiode to a reference value a systematic offset in the output values is present. This gives a per-pixel variation. An example of a column component is the variation of the input bias current in the bias transistor present in each column of the APS. As FPN is added to all frames or images produced by a sensor and is independent of the illumination, it can be easily removed by subtracting a 'dark' frame from the image. It should be noted that the amount of shot noise will increase with a factor $\sqrt{2}$ [12].

A source somewhat similar in characteristics to FPN is PRNU, the variation in pixel response when the sensor *is* illuminated. This variation comes e.g. from non-uniform sizes of the active area where photons can be absorbed. This is a linear effect: when the size of the active area is increased with a factor $x$, the number of photons detected will also increase with factor $x$. This illustrates the multiplicative characteristic of the PRNU: when the illumination increases, the effect of this source increases as well. Another possibility is the presence of non-uniform potential wells giving a varying spectral response. Hence, if the potential well is locally shallow, long wavelength photons may not be absorbed. Therefore, the PRNU is also wavelength dependent. Another possible source of PRNU in CMOS sensors is due to the sense node in the photodiode. This sense node is integrated in the pixel, and as such the room available for this node is small (the sense node in CCD sensors is located off-chip). This results in a small 'capacitor' which capacitance changes with signal level, resulting in a nonlinear response called 'floating diffusion non-linearity'.

The multiplicative nature of the PRNU makes it more difficult to remove this type of non-uniformity, as simply subtracting a frame does not take this illumination dependent nature into account. In principle it is possible to remove the PRNU, or even add the pattern of a different camera [18]. It is also possible to reduce the PRNU inside the camera by a form of non-uniformity correction [19].

FPN together with PRNU form the pattern noise and is always present, though in varying amount due to varying illumination between successive frames and the multiplicative nature of the latter noise source.

There are also noise sources that do not find their origin on the image sensor but are added further down the pipeline, i.e. when the digital signal is processed. The

[1] Some of these characteristics, such as CFA interpolation or Chromatic Aberration can be used for camera *classification* (see [10, 11]), as these features are characteristic for a camera *model*.

[2] Technically, the PRNU is not 'noise' in the usual sense, as it represents a systematic (repeatable) addition to the signal. However, it is customary to denote the following sources as noise.

most obvious source of this type of noise is the quantisation noise, introduced when the analogue information from the sensor (the potential change detected for each pixel) is digitised in the analogue-to-digital converter. This is however a small effect. Another effect that occurs in the processing stage is the demosaicing of the signal. CCD and CMOS image sensors are monochrome devices, i.e. they detect the amount of light incident on each pixel but cannot distinguish the colour of the incident light. To produce colour images a regular pattern in the form of a Colour Filter Array is present above the image sensor, such that only one certain colour (or wavelength range) is absorbed by each pixel. As a result each pixel only records the intensity of one colour, and in this way a mosaic is obtained. To give each pixel its three common RGB values, the colour information of neighbouring pixels are interpolated. This interpolation gives small but detectable offsets, and can be seen as a noise source [20, 21]. Also, dust present on the lens may contribute to the detected pattern noise, as well as possible optical interference in the lens system. As denoising algorithms cannot differentiate between the characteristic PRNU and noise from the processing stages, these effects may also be present in the estimated PRNU pattern.

# 3 Extracting the PRNU pattern

## 3.1 Introduction

We have seen in the previous section that various random and systematic noise sources corrupted the image to a certain extent during acquisition. The goal of a denoising filter is to suppress or remove this noise, without substantially affecting the (small) image details. In general, denoising algorithms cannot discriminate between true noise and small details. It is therefore important to select an appropriate filter that leaves the image structure in tact, most notably around edges where the local variance is high. For example, simple spatial filtering such as the Gaussian smoothing filter [3] removes the noise from an image by low-pass filtering the image data, as noise is generally a high frequency effect. However, as this filter is not able to distinguish between noise and signal features, this method will also distort (blur) the edge integrity. This is a problem especially in images that contain a large amount of details, high frequency textures, etc. The advantage of this filter is that the computational complexity is very low.

The pattern noise in digital images can be seen as a non-periodic signal with sharp discontinuities, as the PRNU is a per-pixel effect. To extract the pattern noise from the image, a wavelet transform is used. The wavelet transform is very similar to a Short-time Fourier Transform (STFT), with some important differences. Instead of a window function we now have a mother wavelet $\psi$:

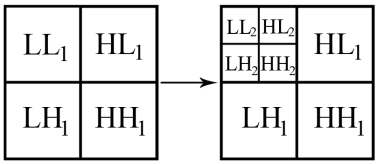


Figure 1: *Subbands of a two dimensional wavelet transform. After the approximation and detail coefficients are calculated, the approximation details ($LL_1$) are split up in high- and low frequency subbands again.*

$$\mathcal{W}f(\tau, s) = \int_{-\infty}^{+\infty} f(t) \frac{1}{\sqrt{s}} \psi^* \Big( \frac{t-\tau}{s} \Big) dt. \qquad (1)$$

where $\mathcal{W}$ denotes the wavelet transform. By scaling and translating this mother wavelet different 'window' functions are obtained, the 'daughter wavelets'. The wavelet transform has an one more parameter compared to the STFT: a translation $\tau$ and a scale $s$. By scaling the mother wavelet the wavelet is dilated or compressed (the 'window' function is resized), and by translating the wavelet the location of the window is changed. A large scale parameter results in a slowly varying daughter wavelet, while a small scale results in a fast varying daughter wavelet. After translating the signal from the beginning of the signal to the end of the signal the wavelet representation for this scale is obtained. The coarsest scale (large $s$, a 'window' with large support) detects low frequencies, the approximation details. On the contrary, a fine scale is sensitive to high frequencies, the detail coefficients, as can be seen from the formula. Each scale represents a different subband, as can be seen in figure 1. The scale and translation parameters are related: when the scale parameter increases, the translation parameter is increased as well. In this way the wavelet functions are localised in space and in frequency, and solve the drawback of the (short time) Fourier Transform. Namely, the Windowed Fourier Transform only uses a single window in which the frequencies are found, while the Wavelet Transform uses variable size 'windows'. The Wavelet Transform is like the Windowed Fourier Transform with variable size window and an infinite set of basis functions. We use a large window for finding low frequency components and small windows for finding high frequency components. Hence, compared to the (Short-time) Fourier Transform, the wavelet transform gives better time resolution for high frequency components, while it gives a better frequency resolution for low frequency components.

By calculating the wavelet coefficients for different values of $s$ and $\tau$, the wavelet representation is obtained. When a wavelet coefficient is large, a lot of signal energy is located at that point, which may indicate important image features such as textures or edges. On the other hand, when a wavelet coefficient is small, the signal does not strongly correlate with the wavelet which means a low amount of signal energy is

present and indicates smooth regions.

We employ the denoising filter as presented by Lukáš *et al* [1], which in turn is based on the work presented in [9], in which an algorithm used for image compression is used for image denoising . An extensive description of the used algorithm follows, and for further details the interested reader is referred to the aforementioned works. The presented algorithm was implemented using the free WaveLab package [24] in Matlab, and has been integrated in the open source NFI PRNUCompare program that is freely available [25].

## 3.2 Algorithm

To perform video camera device identification, the video is first split up in individual frames using FFmpeg [26]. The image is assumed to be distorted with zero-mean White Gaussian Noise (WGN) N(0, $\sigma_0^2$) in the spatial domain with variance $\sigma_0^2$, and hence this noise is also WGN after transforming it to the wavelet domain.

The input frames (images) must be dyadic, as we generally choose base 2 (dyadic sampling) so that the coefficients for scale $2^j$ , $j = 1 \dots n$ are computed. The translation $\tau$ depends on the scale, and can be dyadic as well. The end result is an image with the same size as the input image, composed of nested submatrices each representing a different detail level, as shown in figure 1. This is done for all frames extracted from the video. We want to stress again that calculating the PRNU pattern as presented here can easily be done with the open source NFI PRNUCompare, which is made freely available to the public.

We now present the actual algorithm [1].

1. The fourth level wavelet decomposition using the Daubechies wavelet is obtained by letting a cascade of filters work on the image data, decomposing the image into an orthonormal basis (known as transform coding). We have seen that the Coiflet wavelet may have a slightly better performance. The level-1 approximation coefficients are obtained by filtering the image data through a low-pass filter $g$, while the level-1 detail coefficients are obtained by filtering the image data through a high-pass filter $h$. These two filters are related, in such a way that the original signal can be obtained by applying the filters in reverse ('mirrored') order (these filters are called Quadrature Mirror Filters). By filtering the level-1 approximation coefficients ($LL_1$ subband) with the same set of filters $g$ and $h$, the level-2 approximation and detail coefficients are produced by iteration, as represented in figure 2 (see e.g. Ch. 5 of [27]).

   Each resolution and orientation has its own subband, with $HL_1$ representing the finest details at scale 1 where the high pass filter was applied in the horizontal direction and the lowpass filter in the vertical direction. $LL_4$ represents the low resolution residual.
   This wavelet decomposition into different detail and approximation levels allows the image to be represented as a superposition of coarse and small details, as schematically represented in figure 3.

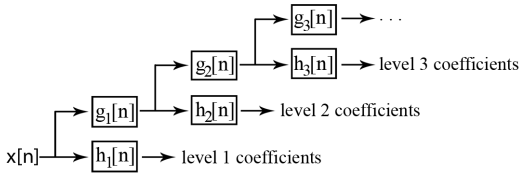


Figure 2: *Iterated filterbank. The output of the lowpass filter g is the input of the next stage. See also figure 1.*

2. For all pixels in each subband the local variance is estimated for each coefficient with a variable size square neighbourhood $N$ with size $W \in (3, 5, 7, 9)$.

$$\hat{\sigma}_W^2(i,j) = \max\Big(0, \frac{1}{W^2} \sum_{(i,j)\in N} LH_s^2(i,j) - \sigma_0^2\Big), \tag{2}$$

   with $(i, j)$ representing the pixel location in each subband. This estimates the local signal variance in each subband, and the minimum variance of each pixel for these varying size neighbourhoods is taken as the final estimate:

$$\hat{\sigma^2}(i,j) = \min(\sigma_{w\in W}^2(i,j)). \tag{3}$$

3. The wavelet coefficients in the detail subbands can be represented by a generalised Gaussian with zero mean [29]. This $\sigma_0^2$ parameter controls how strong the noise suppression will be. When we estimate the reference pattern as well as when we estimate the pattern noise from the (questioned) natural image, we need to set this parameter (denoted $\sigma_{ref}$ and $\sigma_{nat}$ respectively). Ultimately this parameter depends on the image itself (and hence also on the compression) and on the size of the noise.
   The actual denoising step takes place in the wavelet domain by attenuating the low energy coefficients as they are likely to represent noise. This is done in all detail subbands ($LH_s$, $HL_s$, $HH_s$ with $s = 1 \dots 4$) while the low resolution residual $LL_4$ remains unadjusted. The Wiener filter denoises the wavelet coefficients:

$$LH_s(i,j) = LH_s(i,j) \frac{\hat{\sigma}^2(i,j)}{\hat{\sigma}^2(i,j) + \sigma_0^2}. \tag{4}$$

   This approach is intuitive because in smooth regions where the variance is small the coefficients will be adjusted strongly as a disturbance in a smooth region is likely caused by noise. On the other hand, in regions that contain a lot of details

Figure 3: *Left is the low resolution residual. The images are obtained by applying the inverse wavelet transform to the wavelet representation of different scales. Moving to the right more detail is added (lower scales) until the final image is obtained.*

or edges, the variance will be large. Hence, these coefficients are adjusted only marginally, and blurring is avoided. This is also the reason why we select the minimum of the local variance for different sizes of the neighbourhood (step 2).

4. The above steps are repeated for all levels and colour channels. By applying the inverse discrete wavelet transform to the obtained coefficients, the denoised image is obtained. By subtracting this denoised image from the original input image, the estimated PRNU pattern is obtained. As a final step this pattern is zero-meaned such that the row and column averages are zero by subtracting the column averages from each pixel and subsequently subtracting the row averages from each pixel. This is done to remove artefacts from e.g. colour interpolation, as suggested in [4]. Wiener filtering of the resulting pattern in the *Fourier* domain, also suggested in [4], was not applied.

### 3.3 Obtaining the sensor noise patterns and detecting the origin

To determine whether a specific questioned video $V_q$ originates from a certain camera $C$, we first extract the individual frames $I_{q_i}$ ($i = 1 \dots N$, with $N$ the amount of frames in $V_q$) from the video, and subtract the denoised image $I_{d_i}$ from each individual frame:

$$p_{q_i} = I_{q_i} - I_{d_i}, \text{ with } I_{d_i} = \mathcal{F}(I_{q_i}), \tag{5}$$

and $\mathcal{F}$ the filter as described above. Ideally, the resulting pattern should not contain any image residue. In other words, the pattern should closely resemble white noise. To obtain the PRNU pattern, we use the maximum likelihood estimator, as derived in [5]:

$$p_q = \frac{\sum_{i=1}^{N} p_{q_i} I_{q_i}}{\sum_{i=1}^{N} I_{q_i}^2}, \tag{6}$$

with element-wise multiplication implied. In a similar manner the reference patterns $p_{r_j}$ from different cameras with a known origin are obtained by averaging a number of these noise residuals in order to suppress the random noise contributions. However, instead of using images that contain natural content, it is preferred to use a flatfield video $V_f$ from which individual flatfield images $I_{f_i}$ can be extracted that have no scene content and an approximately uniform illumination:

$$p_r = \frac{\sum_{i=1}^{N} (I_{f_i} - \mathcal{F}(I_{f_i})) I_{f_i}}{\sum_{i=1}^{N} I_{f_i}^2}. \tag{7}$$

This is done for multiple cameras, each with its own reference pattern $p_{r_j}$. After all the reference patterns are obtained, the final step is to measure the degree of similarity between the questioned pattern and the reference patterns. We use the total correlation (summed over all colour channels) as the similarity measure in order to find out whether a certain pattern $p_q$ originates from a certain camera $C$. In order to do so, we calculate the correlation between the pattern from the questioned video $p_q$ and the reference patterns $p_{r_j}$. When the correlation of $p_q$ is highest for a certain $p_{r_j}$, we decide the video was most likely acquired using camera $j$.

When obtaining flatfield videos is not feasible (e.g. the camera is not available or broken), it is also possible to use (multiple) natural videos with known origin to obtain the reference pattern.

When the resolution in which the videos have been recorded are lower than the native resolution, binning may occur and attenuate the pattern noise. When this occurs, the pattern to be extracted is much weaker which influences the optimal parameter to be used. The same is expected to be true for compression, as strongly compressed videos are expected to have less of the pattern noise available in the video.

### 3.4 Remarks on the uniqueness of the PRNU

It was mentioned in the introduction that the origins responsible for the existence of the PRNU suggest that the sensor noise patterns from different cameras are unique, though large scale tests are scarce. However, it was observed that the estimated reference patterns from cameras of the same type have a slightly similar sensor noise pattern. This slight similarity was not observed when the patterns from different brands were compared, as can be seen in figure 4. When reference patterns of dissimilar brands are compared, the correlations are centred on 0, i.e. there is no (linear) relationship between the patterns. When patterns from the same brand/model are compared the correlation increases, indicating partly similar patterns. Thus, in practice a thorough test should always include a large number of cameras of the same type. In [30] a large scale test is performed using the sensor fingerprints in the process of device identification. In this paper, images downloaded from Flickr from 6896 cameras from 150 different models are used. It is found that the error rates do not increase when multiple cameras of the same model are used. This may contradict the earlier statement that a slight similarity was found, as in figure 4. However,

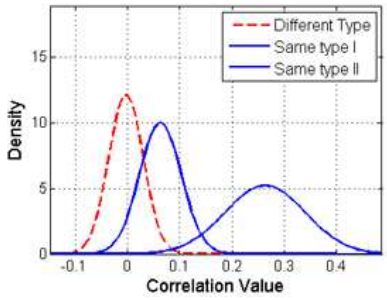


Figure 4: *Correlations between the sensor patterns originating from the same make/brand and correlations between patterns originating from different makes/brands.*

in [30] the cameras are in general of higher quality, so less common artefacts are expected.

Indeed, as already noted in section 2, when some of the artefacts introduced in the output image do not come from the CCD or CMOS sensor itself, but from some other component that is present on all cameras of a certain model and/or type (e.g. a colour filter array), a similarity in the output can be expected. In fact, it is these characteristics that are used in the process of device classification.

In [20] and [31] the camera class is identified by looking at traces left behind by the (proprietary) colour interpolation algorithm that is used to demosaic the colours, after the colour filter array decomposed the incoming light in RGB/CYGM. In [32] different measures are used to train a model with a Support Vector Machine to identify the make/brand of the camera. By using multiple features (high-order wavelet statistics, image quality measures, binary similarity measures) it is possible to a certain extent to identify the make or brand of camera from which an image originates. These characteristics show that other traces left behind in the image are not unique to the individual camera. Hence, device classification shows we need to select an appropriate amount of cameras of the same type and model to compare with.

As the relative size of the individual components responsible for the PRNU is unknown it is possible that the components present on all cameras of the same type contribute a significant amount to the magnitude of the estimated PRNU pattern. For high quality digital cameras this is not a serious problem, as it is expected that these high quality cameras contain less systematic artefacts introduced by compression or demosaicing.

In [4] the problems of non-unique artefacts were removed by zero-meaning the estimates and Wiener filtering the image in the Fourier domain.

# 4 Application to YouTube videos

YouTube is (like Dailymotion, metacafe) a website where users can view and share (upload) video content. Videos encoded with the most popular encoders (such as WMV, DivX, Xvid, but also the 3GP format used by mobile phones) are accepted as upload, after which the uploaded video is converted. Multiple options are available for compressing the videos. Currently, high quality video is offered in 640x480 or 854x480, depending on the aspect ratio of the source video. For standard quality the resolutions and bitrates are lower. Videos are not upscaled; hence if the source resolution is lower than the maximum resolution, the resolution will remain the same. The video encoding is done using the H.264 (MPEG-4 AVC, developed by the Video Coding Experts Group VCEG in collaboration with the Moving Picture Experts Group). The container in which the video presented is either FLV (Flash video) or MP4. If the resolution of the source video is high enough, the video may be viewed in HD quality (1280x720p). Note that unless the video is uploaded in RAW format (that is, unprocessed data, without any form of compression), the resulting video is at least doubly compressed, as the output from the camera may already be a compressed stream.

Online viewing is done using a Flash videoplayer, while downloading these videos can be done using services such as keepvid.com, or the newer keephd.com for downloading HD material. The aspect ratio of the video will generally not change (there are exceptions, see §4.3). As the resolution and the visual quality of the MPEG-4 video is higher (more details are available) than for the Flash video, we use the MPEG-4 video for extracting the pattern noise though the actual bitrate in bits per pixel is lower. To assess the performance of the algorithm for videos that are uploaded to YouTube, we uploaded multiple (natural) videos encoded with different settings and from different cameras to YouTube. The natural videos of approximately 30 seconds were recorded using two popular video codecs, namely Xvid (version 1.1.0) and Windows Media Video 9 (version 9.0.1) in single pass setting, using the Video for Windows (VfW) or DirectShow framework. The WMV9 codec is also used in the popular Windows Live (MSN) Messenger application (see also §4.1.2).

The flatfield video was obtained by recording (without any form of additional compression on top of the possibly compressed output from the cameras) a flat piece of paper under various angles in order to vary the Discrete Cosine Transform (DCT) coefficients in the compression blocks for the duration of approximately 30 seconds. Natural video (also approximately 30 seconds) was obtained by recording the surroundings of the office in which scenes with a high amount of details alternated smooth scenes, both with dark and well-illuminated scenes. Static shots alternated shots with

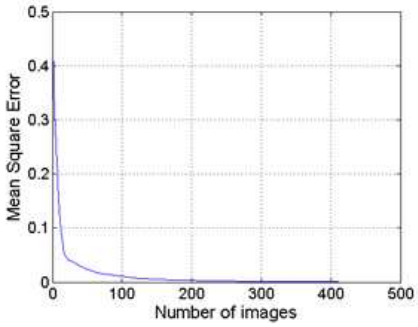


Figure 5: *Mean square error of the estimated pattern noise with respect to the final estimate. We clearly see the estimated pattern converges reasonably quick to the final pattern.*

fast movements, and saturation occurred frequently. All recorded videos have approximately the same content. We made no attempt to select suitable frames based on brightness or other characteristics, other than the removal of saturated frames that occurred at the start of the recording.

When the uploaded (natural) video content has a resolution lower than the maximum resolution from YouTube (1280x720), there is no change in resolution. If this is the case, the reference pattern can be obtained from the RAW video directly from the camera; this gives a better performance compared to uploading the RAW video and finding the reference pattern from the downloaded video.

However, when the resolution of the uploaded (natural) content exceeds the maximum resolution that can be obtained from YouTube, YouTube resizes the input video. As it is unknown how the resizing occur and which artefacts are introduced by the YouTube compression scheme, it is necessary to upload the reference material (in native resolution) to YouTube as well. In this way the reference video undergoes the same processing as the natural video that was uploaded.

Ideally, a large number of frames should be used to calculate the sensor noise patterns. To see how many frames should be averaged we calculated the Mean Square Error (MSE) with respect to the final pattern as obtained from N=450 flatfield frames for the Logitech Communicate STX webcam, see figure 5:

$$MSE(p_{i=1}^{j}, p_{i=1}^{N}), \text{ with } p_{i=1}^{j} = \frac{1}{j}\sum p_i. \qquad (8)$$

We see that the pattern obtained converges quickly to a stable pattern, and that by averaging the patterns from approximately 200 images already a reliable estimate is found. This is not necessarily true for natural video, as the noise estimation depends on the content of the individual frames.

The patterns obtained from each natural video are compared with the reference patterns from all other cameras of the same type.

As explained above, the $\sigma$-parameters control the amount of noise that is extracted from each frame. To see which settings perform best, we calculate the reference patterns as well as the natural patterns (the patterns obtained from the natural video) for multiple values: $\sigma_{nat} = 0.5 + n$, $(n = 0 \ldots 8)$, $\sigma_{ref} = 0.5 + r$, $(r = 0 \ldots 8)$. By calculating the correlation between all these possible pairs we can find the optimum parameters. In actual casework this is not possible, as the questioned video has an unknown origin. We only report the correlation values of the matching (the natural video and reference material have the same origin) and the maximum correlation value of the mismatching pairs (the maximum correlation between the pattern from the natural video and the patterns from all other unrelated cameras), $\rho_m$ and $\rho_{mm}$ respectively. Hence, when $\rho_m$ is larger than $\rho_{mm}$ the correct camera was indeed identified. We tested several cameras, but for reasons of brevity we only report extensively upon one type, namely the Logitech Communicate STX, and only briefly comment on the other cameras and tests.

## Remark

At the time these calculations were done, the maximum available resolution that could be viewed was 480x360 (for aspect ratio 4:3). Recently the maximum resolution that can be viewed (and downloaded) was increased. The experiments were performed when the maximum resolution was still 480x360. Hence, videos that were uploaded as 640x480 could be viewed as 480x360. In the experiments and results that follow we assume this restriction is still in place. This made it necessary to perform additional pre-processing. Namely, when a video was uploaded as 640x480, we could only download it as 480x360. To compare the extracted pattern from the downloaded video with a reference pattern of the same size, it was necessary to resize the reference video, or alternatively upload the reference video in its native resolution to YouTube. For videos with a resolution of 640x480 this is not necessary anymore. However, the same principles hold when a video is now uploaded with a resolution higher than 1280x720.

## 4.1 Logitech Quickcam STX

We recorded for each of the 8 Logitech Quickcam STX cameras a 30 second sample with natural content in the native resolution of 640x480 with the Xvid codec with quality setting 4, as well as a 30 second flatfield sample in the same resolution in RAW. At the time when these calculations were done, this resolution was higher than what could be viewed and downloaded (480x360), hence resizing occurred. As we wanted to compare the noise pattern extracted from these natural videos with the noise patterns extracted from the flatfield videos, we also needed to resize the flatfield videos. This could be done either by uploading the flatfield videos to YouTube, or by simply resizing the individual frames extracted from the reference video using a

bilinear interpolation. The disadvantage of the first option is that the YouTube compression introduces certain artefacts into the output video, which causes a higher correlation between the noise pattern from the natural video and the reference patterns from the flatfield videos. Therefore, it is generally recommended to record the flatfield video in the native resolution of the camera, and subsequently resize the individual frames from the video to match the resolution of the natural material. Of course, when the natural video as downloaded from YouTube has a resolution lower than the maximum allowed resolution (currently 1280x720 or correspondingly for different aspect ratios) this does not need to be taken into account.
Regardless of the parameter settings ($\sigma_{nat} = 4.5 - 6.5$, $\sigma_{flat} = 2.5 - 6.5$ has the best separation), this resulted in a 100% correct identification rate as can be seen in Table 1 (see end of the article).

We also resized the frames from the RAW flatfield video from 640x480 to 480x360 using the bilinear resizing method to match the dimensions obtained from the natural video, and subsequently calculated the correlation between the noise pattern from the natural video and the reference patterns. In Table 2 the results are presented when the RAW flatfield video is resized with the bilinear resizing method.

We repeated the experiment with the same cameras and only changed the recording resolution to 320x240. Recording in a lower than native resolution means in this case that the pixels in the output video are binned (in this case 4 pixels are averaged to give the output of 1 pixel) which results in a strong attenuation of the PRNU, as the PRNU is a per-pixel effect. If one general set of parameters is chosen, a maximum of 6 cameras were correctly identified, as can be seen in Table 3.

#### 4.1.1 Codec variations

For one camera we recorded video in the native resolution of 640x480, as well as the lower resolution 320x240 for two different codecs and different codec-settings. In order to let the video content be the same for all videos, we first recorded the video in RAW at both resolutions, and subsequently encoded it with different codec settings in VirtualDub [33]. For both resolutions we recorded the video in XVID and WMV9, with different codec settings. For the XVID codec we used quality settings $q = 4 \cdot n$, with $n = 1 \ldots 8$, while for the WMV9 codec we used quality settings $q = 10 \cdot n$, $n = 5 \ldots 9$. Note that in the case of XVID higher $q$ values represents higher compression, while in the case of the WMV9 codec a higher setting means higher quality. The videos were uploaded to YouTube, and subsequently downloaded after which the sensor pattern noise was extracted again.

For these settings we again tried to find out whether the outlined method was able to pick out the source camera; a comparison was made with the reference patterns from 7 other Logitech cameras of the same type. For the low resolution 320x240 we used the RAW video to extract the patterns, while for the high resolution it was required to first resize the reference videos.

We see the algorithm performs very well for the 640x480 (native) resolution: the correct identification rate is 100% for all codec settings (Table 4 and 5). Also, the parameter values do not influence the identification rate, and the correct camera is identified for almost all combinations of these parameters.

When the recording resolution is set to 320x240 we see that the correct identification rate is lowered. For the XVID codec we see this happens at the moderate quality setting of 16, while for even lower quality encodings the camera is correctly identified. This shows that video compression is not a linear process; apparently, at lower quality settings more important details are retained. For the WMV9 codec we see the correct identification rate is decreased for the lowest quality settings (Table 6 and 7).

#### 4.1.2 Video extracted from Windows Live Messenger stream

Windows Live Messenger, formerly known as MSN Messenger, is a popular instant messaging client, which provides webcam support as well as videochat support.

Through the use of external programs it is possible to record the video stream sent during a webcam session, often simply by capturing the screen. It is also possible to directly record the data from the stream, as is done with MSN Webcam Recorder [34].

As a final test with this webcam, we set up a webcam session between two computers with Windows Live Messenger, with one computer capturing a webcam stream of approximately two minutes sent out by the other computer. The stream was sent out as a WMV9 video at a resolution of 320x240 (selected as 'large' in the host client). After the data was recorded with the aforementioned program, it was encoded with the XVID codec (1.2.-127) with a bitrate of 200 kbps, which resulted in 1705-1815 frames (0.17-0.18 bpp). Finally, the resulting video was uploaded to YouTube, where a third layer of compression was added. It has to be stressed that in practice with low bandwidth systems the framerate may be reduced significantly.

We again see the source camera is correctly identified.

### 4.2 Creative Live! Video IM

For 6 Creative Live! Cam Video IM webcams we recorded a 30 second sample in 352x288 (11:9), while the native resolution is 640x480 (4:3). These videos were encoded using the WMV9 codec, with quality setting 70 and uploaded to YouTube. This resulted in videos with a bitrate between 180 and 230 kbit/s (0.13-0.16 bpp). Only 5 out of 6 cameras were correctly identified with the optimal parameters of $\sigma_{nat} = 6.5$ and $\sigma_{ref} = 2.5$. Next, we recorded a video with resolution of 800x600 (4:3) and encoded it with quality setting 60 in WMV9. After recording the flatfield videos in the

native resolution, we either have to upload these reference videos to YouTube or resize the individual frames from the flatfield videos. In this case it is advantageous to resize the individual frames: the correlations of mismatching pairs is more closely centred on zero, indicating no (linear) relationship between the noise patterns extracted from the natural video and the patterns extracted from the reference videos. This resulted in a 100% correct identification rate, although the difference between matching and mismatching correlations is low.

### 4.3 Vodafone 710

For each of the ten Vodafone 710 mobile phones, we recorded a 30 second sample in the native resolution of 176x144. This phone stores the videos in the 3GP format. This is, like the AVI file format, a container format in which H.263 or H.264 can be stored. This phone uses the H.263 format optimised for low-bandwidth systems. The natural video had a bitrate between 120 and 130 kbit/s (0.36-0.39 bpp). After uploading the natural videos, YouTube automatically changed the aspect ratio, from 11:9 to 4:3. The correct identification rate is only 50%.

The correct identification rate for this camera is much lower than for the other cameras. This may be due to the codec used to initially encode the source video, namely H.263. This codec uses a form of vector quantization, and is therefore different from the discrete cosine transform used in WMV9 and Xvid.

## 5 Future work

Although video processing is much less common than image processing, a problem occurs when the questioned video has undergone some form(s) of spatial transformations. For example, when videos are cropped, scaled or rotated, the extracted PRNU pattern is transformed in a similar manner. Also, video stabilisation may irreversibly destroy the pattern. Hence, the found pattern will not match the reference pattern acquired from the source camera. For simple transformations like cropping and scaling, doing a brute force search may be possible, as is done for images in [35]. However, for videos this may not be realistic or computationally feasible, as at least a few hundred frames are needed to estimate the PRNU pattern reliably. Doing a brute force search in this case means much more computational power is needed.

As an alternative it may be interesting to see how camera classification performs in this case, for example by means of a Support Vector Machine (SVM). In this approach BSM, IQM, HOWS features are used to build a classifier that is able to distinguish between different camera brands/models. The advantage is that it can easier be made robust to image manipulations, as is shown in [36] for images originating from cellphone cameras. In the aforementioned paper an informed classifier was built, i.e. a classifier that was also trained with manipulated images. Rotations could be detected in this way with accuracy above 80% when 9 cameras were used, depending on the angle over which the image was rotated.

## 6 Conclusion

It was previously shown that the technique used, the extraction of the Photo Response Non-Uniformity, was successfully applied in the source identification of digital cameras and digital scanners. We have seen that it is possible to identify low resolution webcams based on the extraction of sensor noise from the videos it produces, even after these videos were re-compressed by YouTube. Depending on various video characteristics (recording resolution, the codec and bitrate used to encode the video, etc.), reliable identification is possible in a wide range of circumstances. This is not the case for all video material tested: the H.263 compression that is used in the mobile phone we tested does not leave enough pattern noise in the video, and in this case no reliable identification was possible.

We have seen that a change of aspect ratio (e.g. from 4:3 to 16:9) after recording is detrimental to the reliable extraction of the sensor noise. Also, recording videos in non-native resolution may present problems due to the occurrence of binning (e.g. recording a video in 640x480 when the native resolution is 1280x960). Hence, we do not always know in which resolution a video was recorded and/or uploaded.

Another problem is that YouTube may change its encoding scheme from time to time, such that at the time the original (natural) video was uploaded the codec (settings) used to encode the video to H.263 or H.264 may be different compared to when the reference material is uploaded. However, as long as no spatial transformations are applied (such as changing the aspect ratio), this is no severe limitation.

As there are a lot of parameters (duration of the video, content of the video, amount of compression, which codec was used to encode the video, which parameters should be used to extract the noise patterns, with which resolution was the video recorded, etc.) it is not possible to give a general framework to which a video should comply in order for a correct identification to occur. In general, by setting the parameter for extracting the PRNU pattern from natural or flatfield videos between 4 and 6, satisfactory results are obtained. It has to be kept in mind that the noise patterns estimated from cameras of the same make and model may contain similarities. Further pre-processing may reduce these similarities and is likely to improve the results. Especially as the output of webcams or mobile phones may actually be a compressed JPEG stream in which the DCT compression blocks can be clearly distinguished, this may introduce higher than expected correlations based on the sensor noise alone. For this reason it is advised to use a large amount

of cameras of the same brand and model in actual casework. In this way we can see whether similarities between noise patterns comes from the actual sensor noise, or whether it comes from post-processing steps such as CFA interpolation that is common to all camera models of the same brand.

The assumption that the PRNU can be modelled as a white Gaussian noise (either in the spatial or the wavelet domain) is only an approximation to the true distribution of the PRNU, as the multiplicative nature of the PRNU implies that well illuminated areas contain more pattern noise than dark areas. Either the denoising parameter could be made spatially adaptive, or a denoising algorithm could be used that does not make these explicit assumptions about the frequency content in the image, for example a Non-Local means approach [37]. However, although the latter approach performs really well in the case of artificially added white Gaussian noise, the actual performance is often below that of the wavelet filter as presented. Also, the computation time is several times larger compared to the wavelet filters.

In general, we have two opposing predictions for the future with respect to the source video camera identification. On one hand we theorise that the sensor noise is likely to decrease when manufacturing standards increase. This will frustrate the camera identification scheme, as the reliable extraction of the pattern is hindered. On the other hand, a few opposing processes occur simultaneously. First, with increasing resolution the relative PRNU size increases (with the same manufacturing technique) in general as well. Also, the storage capacity of these devices (in mobile phones, but also the maximum allowed video size that can be uploaded to e.g. YouTube) increases. Furthermore, the bandwidth of (mobile) networks increases, allowing faster transfers of these files. Finally, due to the increasing processing power in these electronic devices, more advanced compression schemes become available, possibly retaining more of the pattern noise after compression.

# References


[1] J. Lukáš, J. Fridrich and M. Goljan, *Digital Camera Identification from Sensor Pattern Noise* - IEEE Transactions on Information Forensics and Security, volume 1, pp 205-214; 2006

[2] Zeno J. Geradts, Jurrien Bijhold, Martijn Kieft, Kenji Kurosawa, Kenro Kuroki, and Naoki Saitoh, *Methods for Identification of Images Acquired with Digital Cameras* - Proc. SPIE 4232, 505 (2001)

[3] Erwin J. Alles, Zeno J.M.H. Geradts and Cor J. Veenman, *Source Camera Identification for Low Resolution Heavily Compressed Images* - International Conference on Computational Sciences and Its Applications, 2008. ICCSA 2008, pp. 557-567

[4] M. Chen, J. Fridrich, M. Goljan, *Digital Imaging Sensor Identification (Further Study)* - Proceedings of the SPIE, Volume 6505, pp. 65050P, Security, Steganography, and Watermarking of Multimedia Contents IX; 2007

[5] M. Chen, J. Fridrich, M. Goljan and J. Lukáš, *Determining Image Origin and Integrity Using Sensor Noise* - IEEE Transactions on Information Forensics and Security, Volume 3, Issue 1, pp. 74-90 (2008)

[6] Nitin Khanna, Aravind K. Mikkilineni, George T.C. Chiu, Jan P. Allebach and Edward J. Delp, *Scanner Identification Using Sensor Pattern Noise* - Proceedings of the SPIE, Volume 6505, pp. 65051K (2007), Security, Steganography, and Watermarking of Multimedia Contents IX

[7] Hongmei Gou, Ashwin Swaminathan and Min Wu, *Robust Scanner Identification Based on Noise Features* - Proceedings of the SPIE, Volume 6505, pp. 65050S (2007), Security, Steganography, and Watermarking of Multimedia Contents IX

[8] Mo Chen, Jessica Fridrich, Miroslav Goljan, and Jan Lukáš, *Source Digital Camcorder Identification using Sensor Photo Response Non-Uniformity* - Proc. SPIE, Vol. 6505, 65051G (2007)

[9] M. Kivanç Mihçak, Igor Kozintsev and Kannan Ramchandran, *Spatially Adaptive Statistical Modeling of Wavelet Image Coefficients and its Application to Denoising* - Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc, pp. 3253-3256 (1999)

[10] M.K. Johnson and H. Farid, *Exposing digital forgeries through chromatic aberration* - MM&Sec'06, Proceedings of the Multimedia and Security Workshop 2006, September 26-27, 2006, Geneva, Switzerland, pages 48-55, 2006

[11] L. T. Van, S. Emmanuel, and M. S. Kankanhalli, *Identifying source cell phone using chromatic aberration* - Proceedings of the 2007 IEEE International Conference on Multimedia and EXPO (ICME 2007), pages 883-886, 2007

[12] Gerald C. Holst and T.S. Lomheim, *CMOS / CCD Sensors and Camera Systems* - JCD Publishing and SPIE Press, 2007

[13] K. Irie, A.E. McKinnon, K. Unsworth and I.M. Woodhead, *A Model for Measurement of Noise in CCD Digital-Video Cameras* - Measurement Science and Technology, Vol. 19 (2008), p. 045207

[14] R. Loudon, *Quantum Theory of Light* - Oxford University Press, 2001

[15] Hui Tian, Boyd A. Fowler and Abbas El Gamal, *Analysis of Temporal Noise in CMOS APS* - Proc. SPIE Vol. 3649, p. 177-185 (1999), Sensors, Cameras, and Systems for Scientific/Industrial Applications

[16] K. Salama and A. El Gamal, *Analysis of active pixel sensor readout circuit* - IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, pp. 941-945 (2003)

[17] Abbas El Gamal, Boyd Fowler, Hao Min, Xinqiao Liu, *Modeling and Estimation of FPN Components in CMOS Image Sensors* - Proc. SPIE Vol. 3301, p. 168-177, Solid State Sensor Arrays: Development and Applications II, Morley M. Blouke; Ed., 1998

[18] T. Gloe, M. Kirchner, A. Winkler and R. Böhme, *Can We Trust Digital Image Forensics?* - Proceedings of the 15th International Conference on Multimedia, pp. 79-86, 2007

[19] A. Ferrero, J. Campos and A. Pons, *Correction of Photoresponse Nonuniformity for Matrix Detectors Based on Prior Compensation for Their Nonlinear Behavior* - Applied Optics, Vol. 45, No. 11, 10 April 2006

[20] S. Bayram, H.T. Sencar, N. Memon and İ Avcıbaş, *Source Camera Identification Based on CFA Interpolation* - IEEE International Conference on Image Processing, 2005. ICIP 2005, Vol. 3 pp.69-72

[21] Yangjing Long and Yizhen Huang, *Image Based Source Camera Identification using Demosaicking* - 2006 IEEE 8th Workshop on Multimedia Signal Processing

[22] David L. Donoho and Iain M. Johnstone, *Ideal Spatial Adaptation by Wavelet Shrinkage* - Biometrika, Volume 81, pp. 425-455 (1994)

[23] G. Kaiser, *A friendly guide to wavelets* - Birkhauser Boston Inc (1994), ISBN: 978-0-8176-3711-8

[24] David Donoho, Arian Maleki, Morteza Shahram, *WaveLab 850*: http://www-stat.stanford.edu/∼wavelab/

[25] Maarten van der Mark, Wiger van Houten, Zeno Geradts, *NFI PRNUCompare* http://prnucompare.sourceforge.net

[26] FFmpeg, *Program to record, convert and stream audio and video* http://www.ffmpeg.org

[27] Stéphane Mallat, *A wavelet tour of signal processing* - Academic Press, second edition, 1999

[28] S. Grace Chang, Bin Yu and Martin Vetterli, *Adaptive Wavelet Thresholding for Image Denoising and Compression* - IEEE Transactions on Image Processing, Volume 9, pp. 1532-1546 (2000)

[29] S. Grace Chang, Bin Yu and Martin Vetterli, *Adaptive Wavelet Thresholding for Image Denoising and Compression* - IEEE Transactions on Image Processing, Volume 9, pp. 1532-1546 (2000)

[30] M. Goljan, J. Fridrich, T. Filler, *Large scale test of sensor fingerprint camera identification* - Proc. SPIE, Electronic Imaging, Security and Forensics of Multimedia Contents XI, San Jose, CA, January 18-22, 2009

[31] Sevinc Bayram, Husrev T. Sencar and Nasir Memon, *Improvements on Source Camera-Model Identification Based on CFA Interpolation* - Proc. of the WG 11.9 Intl. Conference on Digital Forensics

[32] Oya Çeliktutan, İsmail Avcıbaş and Bülent Sankur, *Blind Identification of Cellular Phone Cameras* - Proceedings of the SPIE, Volume 6505, pp. 65051H (2007), Security, Steganography, and Watermarking of Multimedia Contents IX

[33] Virtualdub, *A free video capture and AVI/MPEG-1 processing utility* http://www.virtualdub.org

[34] *MSN Webcam Recorder, version 1.2rc7* http://ml20rc.msnfanatic.com/

[35] M. Goljan, J. Fridrich, *Camera identification from cropped and scaled images* - Proc. SPIE, Electronic Imaging, Forensics, Security, Steganography, and Watermarking of Multimedia Contents X, San Jose, CA, January 26-31, 2008, pp. OE-1-OE-13

[36] Oya Çeliktutan, Bülent Sankur, İsmail Avcıbaş, *Blind Identification of Source Cell-phone Model* - IEEE Transactions on Information Forensics and Security, Volume 3, No. 3, September 2008

[37] A. Buades, B. Coll, J.-M Morel, *A non-local algorithm for image denoising* - IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2005, Volume 2, June 2005 pp. 60-65; 2005

| | cam1 | cam2 | cam3 | cam4 | cam5 | cam6 | cam7 | cam8 |
|---|---|---|---|---|---|---|---|---|
| $\rho_m$ | **0.1836** | **0.3013** | **0.1926** | **0.2297** | **0.1731** | **0.1967** | **0.1998** | **0.2581** |
| $\rho_{mm}$ | 0.0526 | 0.0369 | 0.0239 | 0.0362 | 0.0406 | 0.0190 | 0.0283 | 0.0315 |

Table 1: Logitech Communicate STX. Natural videos encoded with XVID quality setting 4 in 640x480 resolution. Flatfield videos uploaded and subsequently downloaded from YouTube, after which the pattern noise was estimated with parameters $\sigma_{nat} = 4.5$, $\sigma_{flat} = 3.5$.

| | cam1 | cam2 | cam3 | cam4 | cam5 | cam6 | cam7 | cam8 |
|---|---|---|---|---|---|---|---|---|
| $\rho_m$ | **0.1334** | **0.2301** | **0.1279** | **0.1818** | **0.1596** | **0.1622** | **0.1515** | **0.2099** |
| $\rho_{mm}$ | 0.0374 | 0.0512 | -0.0009 | 0.0342 | 0.0355 | 0.0290 | 0.0118 | 0.0421 |

Table 2: Logitech Communicate STX. Frames from the uncompressed flatfield videos are resized with a bilinear interpolation to match the size of the natural video downloaded from YouTube (480x360), after which the reference patterns are calculated with parameters $\sigma_{nat} = 6.5$, $\sigma_{flat} = 7.5$.

| | cam1 | cam2 | cam3 | cam4 | cam5 | cam6 | cam7 | cam8 |
|---|---|---|---|---|---|---|---|---|
| $\rho_m$ | **0.1044** | **0.0936** | **0.1090** | 0.0153 | **0.0304** | **0.1044** | **0.0984** | 0.0334 |
| $\rho_{mm}$ | 0.0280 | 0.0616 | 0.0803 | **0.0407** | 0.0245 | 0.0526 | 0.0652 | **0.0648** |

Table 3: Logitech Communicate STX. As the natural video downloaded from YouTube is not resized (320x240), the reference patterns are estimated directly from the uncompressed video with parameters $\sigma_{nat} = 3.5$, $\sigma_{flat} = 5.5$.

| setting | size (kB) | frames | kbit/s | bpp | $\rho_m$ | $\rho_{mm}$ |
|---|---|---|---|---|---|---|
| 4 | 4031 | 519 | 932 | 0.202 | **0.2252** | 0.0693 |
| 8 | 2008 | 519 | 464 | 0.101 | **0.2046** | 0.0901 |
| 12 | 1455 | 519 | 337 | 0.073 | **0.1957** | 0.0678 |
| 16 | 1193 | 519 | 276 | 0.060 | **0.1921** | 0.0690 |
| 20 | 1036 | 519 | 240 | 0.052 | **0.1715** | 0.0471 |
| 24 | 960 | 519 | 222 | 0.048 | **0.1612** | 0.0679 |
| 28 | 889 | 519 | 206 | 0.045 | **0.1798** | 0.0677 |
| 32 | 863 | 519 | 200 | 0.043 | **0.1479** | 0.0579 |

Table 4: Logitech Communicate STX. Video recorded in 640x480 with the XVID codec, variable quality. $\sigma_{nat} = 8.5$, $\sigma_{flat} = 7.5$

| setting | size (kB) | frames | kbit/s | bpp | $\rho_m$ | $\rho_{mm}$ |
|---|---|---|---|---|---|---|
| 90 | 6401 | 508 | 1480 | 0.207 | **0.2852** | 0.0665 |
| 80 | 3013 | 508 | 697 | 0.103 | **0.2084** | 0.0599 |
| 70 | 1994 | 508 | 461 | 0.075 | **0.1972** | 0.0521 |
| 60 | 1459 | 508 | 337 | 0.061 | **0.1666** | 0.0620 |
| 50 | 1210 | 508 | 280 | 0.053 | **0.1773** | 0.0689 |
| 40 | 967 | 508 | 224 | 0.049 | **0.1842** | 0.0586 |

Table 5: Logitech Communicate STX. Video recorded in 640x480 with the WMV9 codec, variable quality. $\sigma_{nat} = 8.5$, $\sigma_{flat} = 5.5$

| setting | size (kB) | frames | kbit/s | bpp | $\rho_m$ | $\rho_{mm}$ |
|---|---|---|---|---|---|---|
| 4 | 2206 | 488 | 535 | 0.859 | **0.1173** | 0.0497 |
| 8 | 1238 | 488 | 300 | 0.429 | 0.0795 | **0.0852** |
| 12 | 949 | 488 | 230 | 0.311 | **0.1115** | 0.0541 |
| 16 | 813 | 488 | 197 | 0.255 | **0.0811** | 0.0608 |
| 20 | 750 | 488 | 182 | 0.221 | **0.1474** | 0.0472 |
| 24 | 703 | 488 | 171 | 0.205 | **0.0935** | 0.0684 |
| 28 | 675 | 488 | 164 | 0.190 | **0.1259** | 0.0531 |
| 32 | 660 | 488 | 160 | 0.184 | **0.1026** | 0.0381 |

Table 6: Logitech Communicate STX. Video recorded in 320x240 with the XVID codec, variable quality. $\sigma_{nat} = 5.5$, $\sigma_{flat} = 4.5$

| setting | size (kB) | frames | kbit/s | bpp | $\rho_m$ | $\rho_{mm}$ |
|---|---|---|---|---|---|---|
| 90 | 3023 | 489 | 734 | 0.859 | **0.0939** | 0.0811 |
| 80 | 1717 | 489 | 417 | 0.428 | **0.1107** | 0.0894 |
| 70 | 1229 | 489 | 298 | 0.310 | **0.1483** | 0.0823 |
| 60 | 953 | 489 | 231 | 0.254 | **0.1001** | 0.0800 |
| 50 | 815 | 489 | 198 | 0.221 | **0.0663** | 0.0481 |
| 40 | 700 | 489 | 170 | 0.204 | 0.0592 | **0.0740** |

Table 7: Logitech Communicate STX. Video recorded in 320x240 with the WMV9 codec, variable quality. $\sigma_{nat} = 6.5$, $\sigma_{flat} = 7.5$

| | cam1 | cam2 | cam3 | cam4 | cam5 | cam6 | cam7 |
|---|---|---|---|---|---|---|---|
| $\rho_m$ | **0.1029** | **0.1361** | **0.0792** | **0.1060** | **0.1010** | **0.0770** | **0.0616** |
| $\rho_{mm}$ | 0.0421 | 0.0383 | 0.0476 | 0.0129 | 0.0459 | 0.0288 | 0.0505 |

Table 8: Logitech Communicate STX. Video (320x240) recorded from webcam stream from Windows Live Messenger (WMV9) and subsequently encoded with the XVID codec. $\sigma_{nat} = 4.5$, $\sigma_{ref} = 2.5$

| | cam1 | cam2 | cam3 | cam4 | cam5 | cam6 |
|---|---|---|---|---|---|---|
| $\rho_m$ | **0.0569** | **0.0242** | **0.0381** | 0.0121 | **0.0294** | **0.1017** |
| $\rho_{mm}$ | 0.0240 | 0.0233 | 0.0262 | **0.0533** | 0.0070 | 0.0526 |

Table 9: Creative Live! Natural video recorded in 352x288, wmv70. $\sigma_{nat} = 6.5$, $\sigma_{ref} = 2.5$

| | cam1 | cam2 | cam3 | cam4 | cam5 | cam6 | cam7 | cam8 | cam9 | cam10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\rho_m$ | 0.0106 | -0.0138 | **0.0913** | **0.0632** | **0.0497** | **0.0339** | 0.0401 | 0.0069 | 0.0291 | **0.0326** |
| $\rho_{mm}$ | **0.0340** | **0.0195** | 0.0713 | 0.0212 | 0.0385 | 0.0306 | **0.0548** | **0.0261** | **0.0558** | 0.0286 |

Table 10: Vodafone 710 (176x144, resized by YouTube) H.263 (no further settings possible). RAW downloaded from YouTube, $\sigma_{nat} = 1.5$, $\sigma_{flat} = 6.5$. Results did not improve when the patterns were estimated from resized flatfield frames.