Digital Camera Identification

Neil Jenkins

4th November 2009

1 The Problem

Forensically linking an image to a particular camera can be very useful. In a court of law, the origin of a particular photo may be crucial evidence in a case against child pornographers or industrial espionage. There are a number of approaches that can be taken, but most have severe issues. Metadata in the header may be lost if the image is saved into a different format and can be easily forged. Police needing to verify the authenticity of their images can use special cameras that embed fragile watermarks or store a hash of the image on a secure memory card, but in most situations the image will have been taken on a normal consumer camera.

What's needed then is a way to link the image data itself to the camera. In this essay I will be examining the method proposed by Lukáš et al. to identify cameras by their unique sensor pattern noise [5]. Previous approaches include an investigation into the use of supervised learning based on a vector of numerical features extracted from the spatial and wavelet domains [4]. However, this only achieved 95% accuracy even in the best case; not good enough for most forensic work. Another research project looked at identifying cameras by pixel defects [2] but these are often eliminated in the post-processing of modern cameras and as there are some cameras without any defects in the sensor, this is again suboptimal.

2 Modelling the Camera

In a digital camera, light passes through a lens and hits an imaging sensor. This is divided into pixels, with a colour filter array in front so that each pixel detects a single colour. The data is then interpolated to get full colour and the resulting signal undergoes further post processing for white balance, colour correction, sharpening and gamma correction. Finally, the image is written out to a memory card, possibly after lossy compression.

There are sources of noise at various stages of the image capturing process. *Shot noise* is caused by the number of photons hitting the sensor for a particular pixel varying by a random amount, modelled by the Poisson distribution. Expensive cameras have larger sensors, so that there are more photons hitting each pixel; this means the the small differences cause less variation so there is less shot noise. *Pattern noise* on the other hand is deterministic and will be approximately the same if multiple pictures of the same scene are taken by the same camera.

Pattern noise consists of two main components. *Fixed pattern noise* is caused by dark currents, the small electric current that leaks from photodiodes in each pixel even when no photons are hitting it. This is an additive noise, often suppressed in better cameras by subtracting a 'dark frame' from every image. It is dependent on exposure time and temperature. *Photo-response non-uniformity noise* (PRNU) is the dominant part of pattern noise. This is mainly caused by pixel non-uniformity (PNU) — different pixels have different sensitivities to light due to imperfections in the silicon and manufacturing process. It is therefore very unlikely for two sensors to have correlated patterns and this noise is not affected by temperature or humidity.

Putting all this together, we can make a mathematical model for the image acquisition process:

$$y_{ij} = f_{ij}(x_{ij} + \eta_{ij}) + c_{ij} + \epsilon_{ij}$$

where y_{ij} is the output of the sensor, x_{ij} denotes the photon counts actually hitting the sensor, η_{ij} is the shot noise, c_{ij} is the dark current, ϵ_{ij} is the additive random noise and f_{ij} is a multiplicative factor close to 1 that captures the PRNU noise. If we can determine **f** we can use it as our fingerprint for the camera. Some astronomical cameras actually attempt to remove **f** from the raw sensor data by a technique known as *flat fielding*; they estimate **f** by averaging images of a uniformly lit scene then divide the pixel data before further processing. However, this is not done in consumer digital cameras so this is not an issue for our intended use case.

3 Identification

The basic algorithm for linking a camera to an image is quite simple. First we calculate the camera reference patterns (essentially an approximation to **f**), then we look for a correlation between each of these patterns and the noise of an image.

The easiest way to calculate an approximation to the camera reference pattern is to average multiple images. To speed up this process we can first remove the scene content using a denoising filter and then average the noise residuals instead. Based on experimentation, Lukáš et al. found a wavelet-based filter gave the best results as it removed the most traces of the scene. The technique also works better with uniformly lit images with no features so we only get noise from the sensor. The larger the number of images we average over, the more we suppress random noise and the impact of any scene data; a minimum of 50 images is recommended.

Once we have established this reference pattern, we can see if there is a correlation with the noise of a particular image. To find the noise, we employ the same trick as before: Use the denoising filter to approximate the noise-free image and subtract this (on a pixel-wise basis) from the original, leaving only the noise residual. We then find the correlation between this noise **n** and a particular reference pattern **r** using the standard formula:

$$corr(\mathbf{n}, \mathbf{r}) = \frac{(\mathbf{n} - \overline{\mathbf{n}}) \cdot (\mathbf{r} - \overline{\mathbf{r}})}{\|\mathbf{n} - \overline{\mathbf{n}}\| \|\mathbf{r} - \overline{\mathbf{r}}\|}$$

By experimentally determining the distribution of this correlation for images taken with a camera and images not taken with that camera we can find a threshold for acceptance and estimate the false rejection rate, subject to an upper bound on the false acceptance rate.

4 Experiment

For my experiment I used images captured with a LogiTech webcam. Images were captured in YUYV format, but we discarded the colour data to get 960×720 px greyscale images. As this was actually a video camera, we could just point it at some white paper and get it to capture a stream of 100 photos from which to calculate the reference pattern. The MATLAB script (presented in Listing 1), follows the algorithm described in Section 3 but uses a simple 3×3 median filter instead of the complicated wavelet-based filter described in the original paper.

Listing 1: Calculating the Reference PRNU Pattern

```
width = 960;
height = 720;
a = zeros( height, width );
for n = [0:99]
    i = im2double( imread( sprintf( 'frame%03d.png', n ) ) );
    a = a + ( i - medfilt2( i, 'symmetric' ) );
end
prnu = a / 100;
save 'prnu-reference' width height prnu;
```

4.1 Experiment 1

I took 5 different images from the same camera and 10 other images from a variety of different cameras, including a high end DSLR as well as several consumer point-and-shoot machines. For each of these, I used the same technique to extract the noise residual and then calculated the correlation with the reference pattern for the webcam, using the MATLAB code in Listing 2.

The results are presented in Figure 1. There is an order of magnitude difference in the correlation with the reference image between the images from the same camera and the others. Remembering the original researchers found better results with a more complicated denoising algorithm, we can look at this as a lower bound on performance. This is obviously not an extensive evaluation and we can't use it to

Listing 2: Calculating the Correlation with the Reference Pattern

```
load 'prnu-reference.mat';
% Make it a flat vector rather than a matrix
prnu_vector = reshape( prnu, 1, numel( prnu ) );
% Calculate the mean PRNU value
mean_prnu = mean( prnu_vector );
p = prnu_vector - mean_prnu;
% Look for correlation with each test image
cor = [];
for n = [1:15]
    image = im2double( imread( sprintf( 'testimg-%02d.png', n ) ) );
    image_vector = reshape( image - medfilt2( image ), 1, numel( prnu ) );
    mean_image = mean( image_vector );
    i = image_vector - mean_image;
    correlation = ( i * ( p' ) ) / ( sqrt( i * i' ) * sqrt( p * p' ) );
    cor( n ) = correlation;
end
cor % Print list of correlations
```

calculate any clear thresholds, but it shows that the method definitely works and has the potential to differentiate very accurately between images taken by a particular camera and images not captured by it.

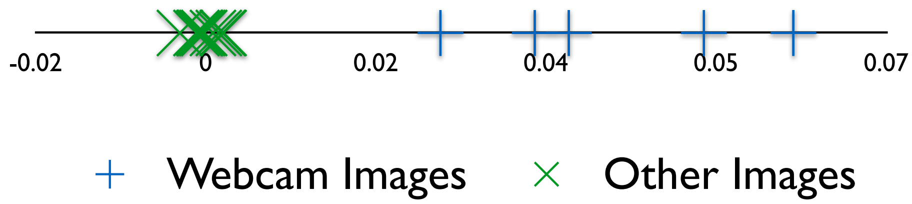

Figure 1: Correlation Values for Images from the Same and Different Cameras

4.2 Experiment 2

To simulate the effect of correlating with a large database of reference patterns, we also looked at cross-correlating a portion of an image taken by the webcam with its reference pattern, using the MATLAB code in Listing 3.

Viewing the result as a greyscale image it is a fairly uniform noisy grey (see Figure 2), except for a single white peak where the image is aligned in the correct place with the reference pattern (i.e. it is in the position it was cropped from); see the zoomed image in section in Figure 3. This is an indicator that the method is quite

Listing 3: Cross-Correlating a Portion of the Image With the PRNU Reference

```
load 'prnu-quickcam.mat'
prnu = single( prnu( 1:720, 1:720 ) );
i = single( imread( 'image.png' ) ); % Read image
i = i( 100:399, 100:399 );           % Extract portion
n = i - medfilt2( i );               % Find noise residue
x = xcorr2( n, prnu );               % Cross-correlate
imagesc( x );                        % And display the result
```

reliable; there is only likely to be significant correlation with the reference pattern from the camera that actually took the image.

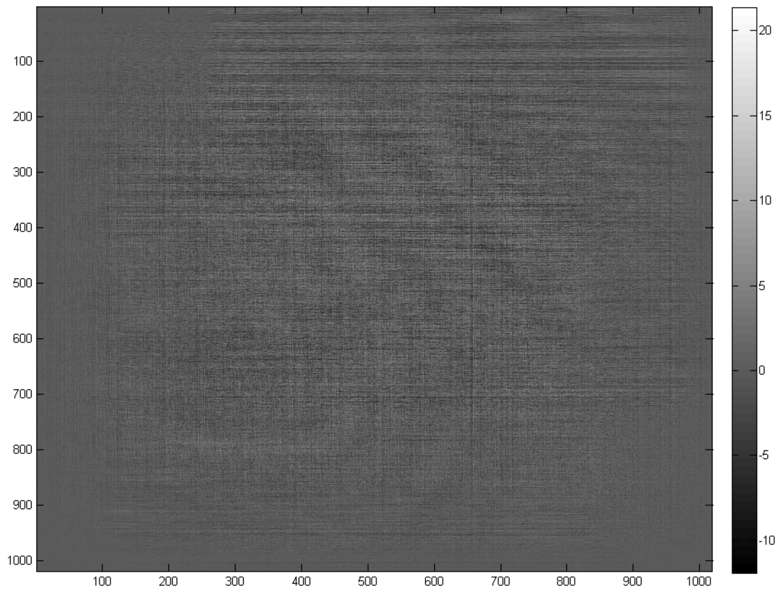

Figure 2: Cross-Correlation of Image Slice With Reference Pattern

5 Results

The authors of the original paper [5] evaluated this technique using 9 different cameras, including two of the same type and taking around 320 images from each. The pictures were taken over a span of 5 years, under a wide range of temperature, humidity and lighting conditions and saved into a variety of formats. The reference pattern for each camera was calculated from 300 images.

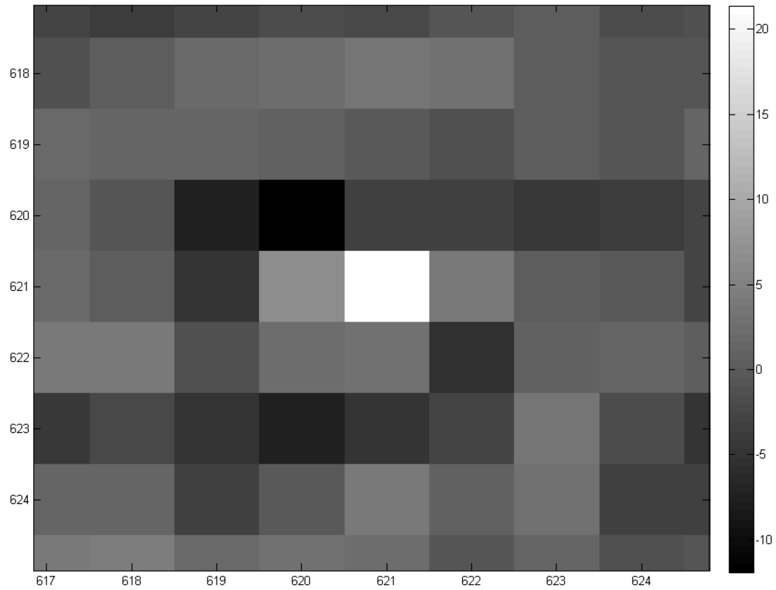

Figure 3: The Single Correlating Position

For every image they tested, the correlation was always highest with the reference pattern for the camera which took the image and the distributions were always well separated. Using a generalised Gaussian model fit, they estimate the false rejection rates for each camera given a false acceptance rate of 10^{-3} and find them to vary between 4.68×10^{-3} in the worst case down to less than 10^{-10} in the best. Further experiments show that gamma correction has very little impact on the reliability of identification and compression into JPEG actually resulted in some cameras getting slightly more accurate results; they hypothesise that the compression removes the small positive correlations between the reference patterns, which may be due to cameras having similar sensors. They look at images taken over 5 years from a camera to show that the patten noise remains stable over time.

6 Improved Model and Algorithm

Further research into digital camera identification was carried out by most of the researchers who worked on the original paper. The result, presented in a later paper [1], is a more theoretically sound method plus a number of post-processing tweaks to improve accuracy.

The new method is based upon a more accurate sensor model:

$$\mathbf{I} = g^{\gamma} \cdot [(\mathbf{1} + \mathbf{K})\mathbf{Y} + \Lambda + \Theta_s + \Theta_r]^{\gamma} + \Theta_q$$

where $\mathbf{I}$ is the signal recorded by the sensor, g is the gain, γ is the gamma correction factor (typically $= 1/2.2$), $\mathbf{K}$ is a zero-mean multiplicative factor (the PRNU pattern), $\mathbf{Y}$ is the light intensity, Λ is the dark current noise, Θ_s is the shot noise, Θ_r is the read out noise and Θ_q is the quantisation noise (caused by having finite bits to represent an analog voltage digitally). From this they derive a maximum likelihood estimator for the PRNU value $\mathbf{K}$; rather than just averaging the residuals, they start with a linearised model of the camera output and formulate the problem as parameter estimation in noisy observations, assuming the corrupting noise sources are Gaussian. Detecting a match is treated as binary hypothesis testing. The image is divided into blocks, each of which is fed into a correlation predictor so that the acceptance threshold required for a correlation is adaptive to the image content.

Before testing the estimated $\mathbf{K}$ for correlation with images, the authors preprocess it to remove some of the components due to colour interpolation, JPEG compression and on-sensor signal transfer. Unlike the PRNU, these are not unique to a particular camera, therefore they may cause increased false positives by increasing the correlation of the reference pattern with images from other cameras. To remove the effect of CFA colour interpolation, they zero out the means of rows and columns in the estimated PRNU; this is done by subtracting the column averages from each pixel in a each column then doing the same for the rows. This also removes a linear pattern introduced by the row-wise/column-wise operation of sensors and image processing circuits. To remove JPEG blockiness and remaining periodic patterns, they transform the zero-meaned PRNU estimation into the Fourier domain, subtract a Wiener filtered version of this and then apply an inverse Fourier transform. These techniques reduce the correlation between reference patterns for different cameras by up to 2 orders of magnitude.

7 Large scale test results

To evaluate the reliability of this improved algorithm, Goljan et al. conducted an extensive test [3] using over a million images drawn from the Flickr image database. The images came from over 6,800 different cameras encompassing 150 different camera models. The camera fingerprints were calculated from 50 random images taken by that camera.

Because they didn't have physical access to the cameras being tested, they couldn't take reference images of neutral scenes. Also, some images may have had digital zoom or other post-processing applied, therefore we can treat this data as an upper bound on the error rates. Despite this, the false rejection rate was estimated to be less than 0.0238 with a false acceptance rate below 2.4×10^{-5}. Testing 145 random images, each from a different camera model, against the database of 6,827 fingerprints correctly identified the source camera in all but four cases.

8 Conclusion

The techniques described in this essay summarise a novel approach to digital camera identification, based on a strong theoretical foundation with good empirical evidence supporting its reliability. Forensic signal analysis will always be a game of cat and mouse, with new techniques sure to be developed to remove or even forge PRNU fingerprints on images. For now, though, we have a fairly reliable method for linking images to a source camera.

References

[1] Mo Chen, Jessica Fridrich, and Miroslav Goljan. Digital imaging sensor identification (further study). In *Security, Steganography, and Watermarking of Multimedia Contents IX. Edited by Delp, Edward J., III; Wong, Ping Wah. Proceedings of the SPIE, Volume 6505, pp. 65050P (2007).*, volume 6505 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, feb 2007.

[2] Zeno J Geradts, Jurrien Bijhold, Martijn Kieft, Kenro Kuroki, and Naoki Saitoh. Methods for identification of images acquired with digital cameras, 2001.

[3] Miroslav Goljan, Jessica Fridrich, and Tomáš Filler. Large scale test of sensor fingerprint camera identification. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7254 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, feb 2009.

[4] Mehdi Kharrazi, Husrev T. Sencar, and Nasir D. Memon. Blind source camera identification. *Proc. ICIP'04*, pages 24–27, 2004.

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