Bangladesh University of Engineering and Technology

A Report on

Source Camera Identification

Submitted to

Shoili Chakma
&
Md. Mukhlasur Rahman Tanvir

Department of EEE

Bangladesh University of
Engineering and Technology

Submitted by

Name	ID
Sohan Salahuddin Mugdho	1606099
A.S.M Sadman Sakib	1606100
Sabit MD Abdal	1606119
Ankan Ghosh Dastider	1606120

Introduction:

Blindly determining the make and model of an image's source camera is an important forensic problem. Information about an image's source can be used to verify its authenticity and origin. This is particularly important since digital images are often used as evidence during criminal investigations and as intelligence in military and defense scenarios. Additionally, source camera identification techniques can be used to uncover similarities between different camera's internal processing, thus potentially exposing intellectual property theft [1]. Many forensic techniques have been proposed to perform camera model identification [2]. Though some of these perform identification using a set of heuristically designed features [3], the majority operate by building a parametric model of a camera component or the artifacts it leaves behind, then using an estimate of these model parameters to identify the source camera model. Techniques have been proposed to identify the model of an image's source camera using

Figure 1. Source Camera Identification

models of a camera's demosaicing algorithm [1], [4], imaging sensor noise [5], lens induced chromatic aberration [6], [7], and proprietary implementations of JPEG compression [8]. Much of the research aimed at increasing the performance of these techniques' performance has focused on improving these parametric models. Most components in a camera's processing pipeline, however, are complex and highly nonlinear. This makes it extremely difficult, if not impossible, to build parametric models that accurately capture intricate characteristics of these components. Recently, Fridrich et al. developed a novel method of dealing with a similar problem in steganography. Rather than attempting to build accurate parametric models of cover and stego images, Fridrich et al. proposed building a rich model by grouping together a diverse set of simple submodels [9]. In this paper, we propose a new framework for identifying the model of an image's source camera. Our framework builds a rich model of a camera's demosaicing algorithm by grouping together a set of submodels. Each submodel is a non-parametric model designed to capture partial information of the demosaicing algorithm. By enforcing diversity among these submodels, we form a comprehensive representation of a camera's demosaicing algorithm.

Background:

When a digital camera captures an image, light reflected from a real-world scene passes through the camera's lens and optical filter before hitting the imaging sensor. Since most cameras are equipped with only one sensor, they cannot simultaneously record all three primary colors of light at each pixel location. To solve this dilemma, most commercial cameras place a color filter array (CFA) immediately before the sensor. The CFA allows only one color component of light to pass through it at each position before reaching the sensor. As a result, the sensor records only one color value at each pixel location. Next, the two unobserved color values at each pixel location must be interpolated using a process known as demosaicing. There are generally two types of demosaicing algorithms: non-adaptive and adaptive. Non-adaptive demosaicing algorithms apply a uniform strategy to interpolate unobserved colors throughout the whole image. Most of modern cameras, however, employ adaptive demosaicing algorithms which can provide higher picture quality. In order to prevent blurring artifacts in textured regions, adaptive algorithms interpolate missing colors in a manner that varies according to the image content. They may also adopt different strategies in different color channels, or interpolate one color

channel using the pixel values of other channels. This will introduce complex intrachannel and inter-channel dependencies, making the demosaicing algorithm very nonlinear. After demosaicing, the image often undergoes a set of post-processing operations such as white balancing, gamma correction, and JPEG compression. A complete overview of a camera's image processing pipeline is shown in Fig. 1. Though the processing pipeline of virtually all digital cameras are composed of the same components, the implementation of each component typically varies from manufacturer to manufacturer, and from model to model. Furthermore, many of these components leave behind traces in an output image. As a result, many forensic techniques have been developed that use these traces to identify the make and model of an image's source camera. Most of these techniques roughly operate by developing a parametric model of a specific component, or the trace it leaves behind. Next, these parameters are estimated for each image on a large training database of images captured by a variety of different camera models. Finally, these parameter values are used as features to train a classifier to identify an image's source camera. A significant amount of previous camera model identification research has focused on two components: the CFA pattern and demosaicing algorithm. In [1], Swaminathan et al. jointly estimate the CFA pattern and the demosaicing filter /coefficients, then use these coefficients as features to train a support vector machine to determine the source camera model. To do this, they assume that the color interpolation algorithms are local linear in different textural regions. In their algorithm, images are divided into three different regions according to the gradients, and the color interpolation parameters of each region are estimated linearly. The CFA pattern is determined at last by choosing the candidate CFA pattern that yeilds the smallest reinterpolation error. Cao and Kot develop a partial second-order derivative correlation model to formulate the demosaicing process [4]. They divide all demosaiced color components into 16 categories and build a set of linear demosaicing equations from the partial derivative correlation model for every category. An expectation-maximization reverse classification (EMRC) algorithm is applied to estimate the

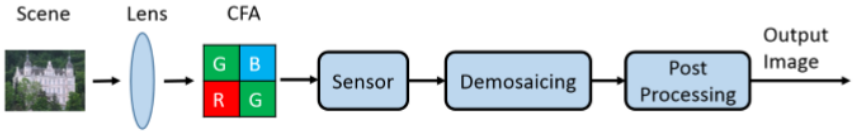

Figure 2. The processing pipeline in a digital camera

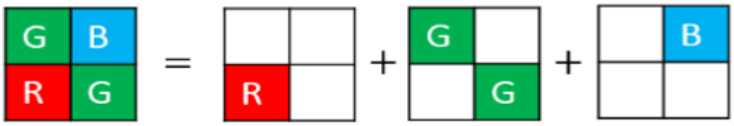

Figure 3. The Bayer pattern

demosaicing weights for each category. Finally, estimated weights, error statistics and category sizes are used as features for classification. While both of these algorithms can achieve good performance, they are limited by the fact that both essentially utilize linear or local linear parametric model of the demosaicing process. As we mentioned above, modern demosaicing algorithms are both non-linear and adaptive, and contain complexities that are difficult to capture using these linear models. Furthermore, these complexities may be difficult or impossible to accurately represent even using sophisticated parametric models. While this poses a difficult challenge for camera model identification research, it does not mean that these deomsaicing algorithms cannot be accurately represented. In the next section, we propose a new method to accurately capture the effects of demosaicing algorithms for camera model identification.

Source Camera Identification Framework:

In our proposed framework, we avoid building parametric models and estimating model coefficients because it's difficult to accurately approximate the components in real cameras. We use another way to represent the CFA pattern and color interpolation algorithm inspired by Fridrich and Kodovsky's work in [9]. Fridrich and Kodovsky developed an universal strategy for steganalysis. They first predict pixel values based on the neighboring pixels with various prediction filters. Under the assumption that natural images are smooth and noise in images is independent of content, the prediction error of stego images which contains hidden information embedded by some steganography techniques will present a different statistical characteristic compared to the error of authentic images. They design a set of diverse submodels to represent the joint probability distribution for each type of prediction error. Each submodel can capture a slightly different trace left by embedding algorithms. Hence, the rich model consisting of diverse submodels can provide powerful information about the embedding algorithms. We adopt this approach for our source camera identification problem. We build a rich model of demosaicing algorithm used in a camera by generating a diverse set of submodels. Each submodel can only capture part of information about the demosaicing algorithm. To obtain an overall picture of the demosaicing algorithm of a camera, we enforce the diversity of submodels by designing different ways to generate them so that every submodel conveys different aspects of demosaicing

information. Thus with a large number of diverse submodels, the rich model grouped from them can yield a much more comprehensive representation of the sophisticated, non-linear color interpolation strategy in cameras compared to mathematical parametric models. We then form our feature space by merging all submodels together and feed them into a multiclass ensemble classifier for camera model identification.

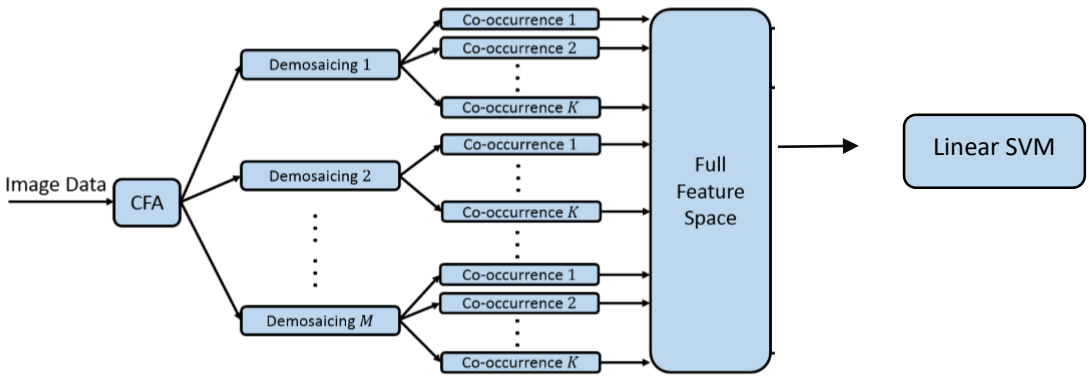

Figure 4. Architecture of our camera model identification framework

We reconstruct image data of a camera by re-sampling color components according to the CFA pattern and re-interpolating missing colors with M preselected baseline demosaicing algorithms. We then end up with M demosaicing errors which are the differences between the original images and our reconstructed versions. For every demosaicing error, we design K different geometric structures and build submodels by calculating co-occurrence matrices over each geometric structure. The co-occurrence matrix is essentially a way to represent the joint probability distribution of demosaicing errors. A more detailed description of cooccurrence matrix is presented later in this paper. The geometric structure of the joint error distribution can take both different color layers and relative positions in the assumed CFA pattern into account.

Thus the M×K submodels formed from K geometric structures and M baseline demosaicing errors can capture both intra-channel and cross channel correlation of camera's demosaicing algorithm. We now describe the details of how we implement our proposed framework.

Feature Collection

1) Re-sampling and Re-interpolation: Among all CFA patterns, the Bayer pattern is the most commonly used. In this paper, we assume all the cameras we inspect employ the Bayer pattern shown in Fig. 2. Since only one color is recorded at one pixel, the blank blocks denote the missing colors in every channel which have to be interpolated. For a particular image X = {R,G,B} where R,G,B represent red, green and blue channel of X respectively, we apply demosaicing process Demos with re-sampling CFA pattern denoted as CFA and baseline interpolation algorithm H. Then the demosaicing error E is obtained as follows.

$$E = X - Demos_{CFA,H}(X) \qquad (1)$$

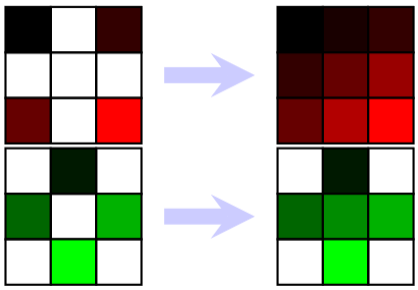

Figure 7. Bilinear interpolation of the red and green channels

In the demosaicing process, we re-sample the pixel values in three channels according to the chosen CFA pattern. The re-sampled pixel values are supposed to be directly captured by the sensor and have no errors. Then, the other missing two color components have to be domosaiced with nearest neighbor interpolation, bilinear interpolation or other content adaptive demosaicing algorithms. The error is calculated as the difference between reconstructed image and original one.

1.1 Interpolation: Since our target is a full-colour, full-size image, digital cameras have to interpolate the pixel values across each of the three colour channels. In this section, I describe several such interpolation algorithms. There are many different interpolation models used in

commercial digital cameras, and an exhaustive list would be unfeasible—however most are similar in effect or principle to the following (as described in Popescu and Farid[3]). When describing the algorithms, let Sx,y be the CFA image1, and let $\hat{}\, Rx,y$, $\hat{}\, Gx,y$, $\hat{}\, Bx,y$ be the red, green and blue channels constructed from Sx,y, where

$$\hat{}\, Rx, = \{Sx,y \text{ if } (x \equiv 1 \bmod 2) \wedge (y \equiv 1 \bmod 2) \; 0 \text{ otherwise} \quad (1)$$

$$\hat{}\, Gx, = \{Sx,y \text{ if } (x \equiv 1 \bmod 2) \oplus (y \equiv 1 \bmod 2) \; 0 \text{ otherwise} \quad (2)$$

$$\hat{}\, Bx, = \{Sx,y \text{ if } (x \equiv 0 \bmod 2) \wedge (y \equiv 0 \bmod 2) \; 0 \text{ otherwise} \quad (3)$$

Then, let $Rx,$, Gx,y, Bx,y be the colour channels of the target, interpolated image. It is these three that we want to calculate. In all the below cases, my Matlab implementation can be found in appendix A, and example outputs of the Matlab implementations can be found in appendix B.

1.1.1Nearest-Neighbour Interpolation : In nearest-neighbour interpolation, the empty pixel values are filled in with their nearest neighbour's value. An example of such an interpolation is:

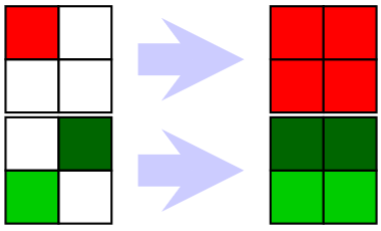

Figure 5. Nearest-neighbour interpolation of the red and green channels

$$R2i,2j = R2i+1,2j = R2i,2j+1 = R2i+1,2j+1 = \hat{}\, R2i+1,2j+1 \quad (4)$$

$$G2i,2j = G2i,2j+1 = \hat{}\, G2i,2j+1 \quad (5)$$

$$G2i+1,2j = G2i+1,2j+1 = \hat{}\, G2i+1,2j \quad (6)$$

$$B2i,2j = B2i+1,2j = B2i,2j+1 = B2i+1,2j+1 = \hat{}\, B2i,2j \quad (7)$$

Figure 6. demonstrates the above algorithm visually.

Such an interpolation causes unpleasant blocky artefacts, and is not generally used unless a very high-speed implementation is necessary.

1.1.2 Bilinear Interpolation: Bilinear interpolation estimates the value of an empty pixel as the average of the values of its nonempty neighbours: more precisely, it performs a convolution of the matrix representing each channel with an bilinear interpolation matrix. Therefore, we have:

$$R = \hat{R} * \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{8}$$

$$G = \hat{G} * \frac{1}{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{9}$$

$$B = \hat{B} * \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{10}$$

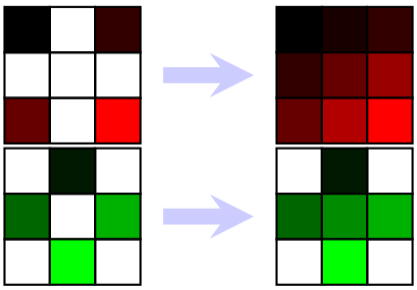

Figure 7. Bilinear interpolation of the red and green channels

Other forms of linear interpolation using convolutions also exist, including bicubic and biquadratic, however bilinear is the most common.

2) Quantization and Truncation: After demosaicing with various base line demosaicing algorithms,we get a set of different demosaicing errors. Every error is a three-layer matrix and pixel values in the positions which should be directly observed by the Bayer pattern are all zeros.

If we want to use cooccurrence matrices to approximate its empirical probability distribution, we have to control the value and range of the error. Therefore, we quantize it with step q and truncate it with threshold T.

$$E \leftarrow \text{trunc}_t(\text{round}(E/q)) \qquad (2)$$

Wechoose q = 2 and T = 3 tobuildco-occurrencematricesin our experiment. However, the quantization step and truncation threshold are not unchangeable. In fact, we believe that by closely examining the distribution of demosaicing errors, there should be more efficient and adaptive way to decide their values.

3) Co-occurrence Matrix: We want to find the statistical property of demosaicing errors by building co-occurrence matrices (i.e. each submodel) which are an approximation of the joint probability distribution of error values on designed geometric patterns. The reason why we don't treat error value in every position independently is that there are dependencies not only among errors on different positions within one channel but also among errors in different channels due to the adaptive property of real demosaicing algorithms. Cooccurrence matrices built within and between color channels can capture these intra-channel and inter-channel dependencies which tell us information about demosaicing strategy in cameras. The format of our demosaicing errors provide us with flexibility to build co-occurrence matrices. Ignoring pixels in every channel which are directly observed according to the CFA pattern, we can design a lot of geometric structures to calculate diverse co-occurrence matrices from demosaiced pixel values. Each co-occurrence matrix conveys its own part of statistical characteristics of the demosaicing error. Here we show an example of generating co-occurrence matrix within

red channel in (3) and (4) where G1, B, R and G2 are the sets of pixel locations which supposed to be directly observed by the Bayer pattern. C(R)CFA,H(d1,d2,d3) denotes the cooccurrence of red channel with CFA and H as the assumed CFA pattern and demosaicing algorithm. |·| is the cardinality of a set and 1(·) is the indicator function. We count frequency of the triple (d1,d2,d3) appearing in the geometric format

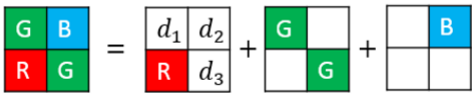

Figure 8. An example of geometric structure to build co-occurrence matrix of red channel.

shown in Fig. 8 as the joint probability distribution of the three demosaiced values of red channel within the Bayer pattern

$$G1 = \{(i,j)|i \text{ odd}, j \text{ odd}\}$$

$$B = \{(i,j)|i \text{ odd}, j \text{ even}\}$$

$$R = \{(i,j)|i \text{ even}, j \text{ odd}\}$$

$$G2 = \{(i,j)|i \text{ even}, j \text{ even}\} \quad (3)$$

$$\mathbf{C}^{(R)}_{CFA,H}(d_1,d_2,d_3) = \frac{1}{|\mathcal{G}1|}\sum_{(i,j)\in\mathcal{G}1}\mathbb{1}\Big((\mathbf{R}_{i,j},\mathbf{R}_{i,j+1},\mathbf{R}_{i+1,j+1}) = (d_1,d_2,d_3)\Big) \quad (4)$$

After calculating all designed co-occurrence matrices for all demosaicing errors, we merge (unite) these matrices as the feature set for classification. In our experiment, we only design two geometric patterns to generate co-occurrence matrices because the more co-occurrences we have, the higher dimension of features we get. Due to the curse of dimension in machine learning, overfitting problem easily happens under the circumstances of high dimensional features and insufficient data. Given that it is not feasible gather a tremendous amount of data for every camera model to overcome this problem, dimension of feature is controlled by using a small number of co-occurrences. In the experimental results, we will show that the two designed co-occurrences can still capture an effective amount of demosaicing information to distinguish different camera models.

Training

The Classification Learner app lets you train models to classify data using supervised machine learning.

Using Classification Learner, you can perform common machine learning tasks such as interactively exploring your data, selecting features, specifying validation schemes, training models, and assessing results. Choose from several classification types including decision trees, support vector machines (SVM), and k-nearest neighbors, and select from ensemble methods such as bagging, boosting, and random subspace.

Classification Learner helps you choose the best model for your data by letting you perform model assessment and model comparisons using confusion matrices and ROC curves. Export classification models to the MATLAB workspace to generate predictions on new data, or generate MATLAB code to integrate models into applications such as computer vision, signal processing, and data analytics. We used the data set of Kaggle for training. We took 150 pictures for each model and 25% is hold for testing for our project of source camera identification. There was 686 identifier for the classification.

Result

if we give the input picture the result is shown in the comman window and it shows the model of the source through which the photo was taken. The accuracy of the result is almost 90-95%.If we can make the code more efficient the error will be lesser and we will get more accurate results. The picture of the output is given below:

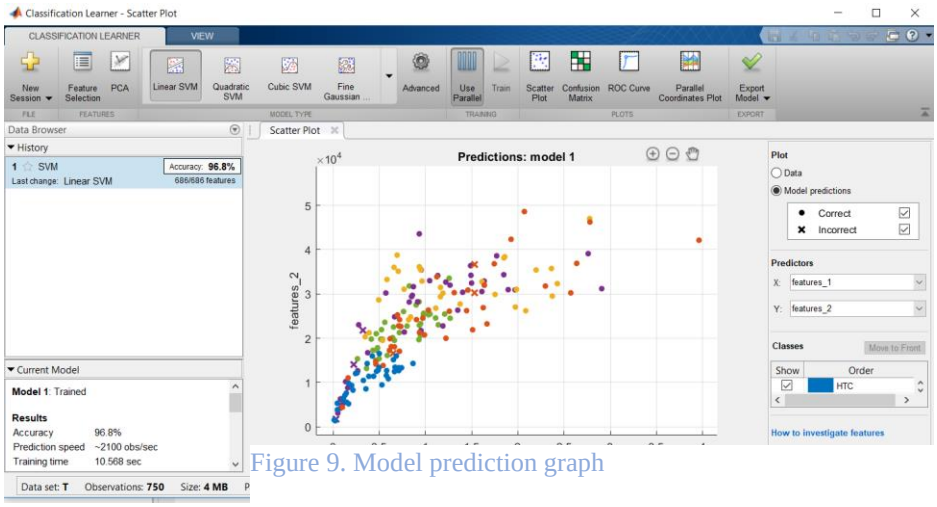

Figure 9. Model prediction graph

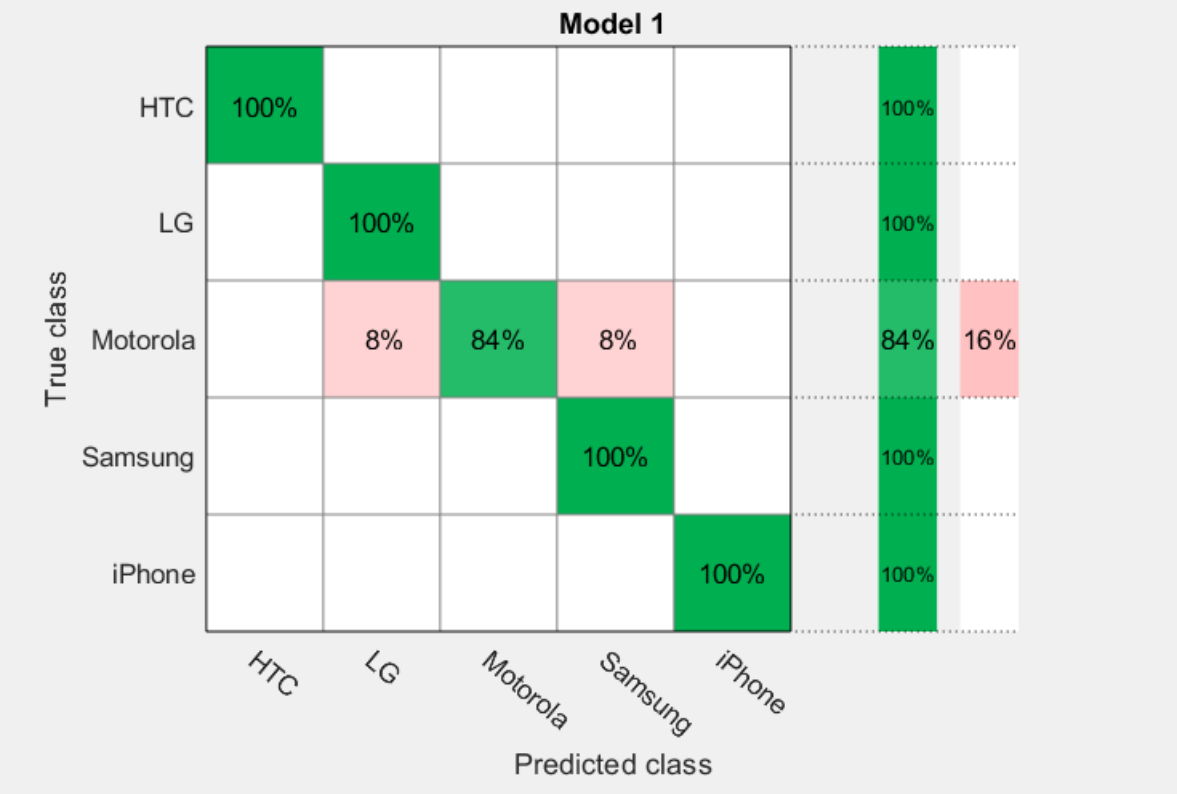

Figure 10. True class vs predicted class graph

Command Window

```
8×1 cell array

    {'Samsung' }
    {'HTC'     }
    {'LG'      }
    {'LG'      }
    {'Samsung' }
    {'Motorola'}
    {'iPhone'  }
    {'iPhone'  }
```

2 labels
'(GalaxyS4)151.jpg'
'(HTC-1-M7)174.jpg'
'(LG5x)152.jpg'
'(LG5x)203.jpg'
'(MotoMax)176.jpg'
'(MotoMax)178.jpg'
'(iP6)252.jpg'
'IMG_5025.JPG'

Figure 11. The output