# Selected Topics

### 20 Feb 2020 (Digital Forensics)

#### Digital Forensics

Digital Forensics is regarding all the possible actions describing manipulation of multimedia digital objects. In the field, there are very very specific specializations.

#### Methodology can be Active or Passive

Active is considered requiring some sort of altering or manipulation of the digital media (i.e: cryptography, etc) and you require an entire cryptographic structure. Watermarking would fall under the active method, and can be thought of as fragile or robust, depending on whether compression affects the watermark)

Passive is considered not altering or manipulating the digital media, but instead, comparing the source identification, as well as the integrity of the media.

Therefore it is clear from the description that these two things solve slightly different problems or concerns within the field.

#### Digital Media has digital fingerprints

This concept is akin to the traditional modes of forensics, which thinks about how human beings leave behind traces, such as fingerprints that identify themselves in an act. This concept applies in a way to digital media through the concept of camera lenses as well as other factors. The digital fingerprint is broken down into three categories:

1. In-camera fingerprints (this would be your lens imperfections, etc)

2. Out-camera fingerprints (this would be your post-processing signature, etc)

3. Geometric (scene) fingerprints (this would be the physical environment, lighting, shadows, reflections, abberations, etc)

#### Image Acquisition can be broken down into eight parts

We can think about this image acquisition process, and identify how we can analyze it.

1. Light projection to some sensor

2. mapping values from sensors -> digital process

3. adjusting signals

4. filters to account for colors

5. interpolation/demosaiking

6. color correction/gamma correction

7. sharpening/post-processing

8 storage of images

#### Different sensors (cameras) have different unique properties

We can break these properties down by the noise level:

1. Fixed Pattern Noise (this would be due to micro-imperfection of the lenses, causing dead pixels etc... on a scale of a 100 megapixel camera, it is almost impossible to replicate the imperfections exactly.)

2. Photo-response non-uniformity (this is more like sensors not responding to the environment in exactly the same way, meaning that certain pixels may have slightly different representations between sensors. If you think about the pixels or sensors as buckets, uniform light should yield uniform buckets, but this is of course not the case)

#### We can detect manipulation through a concept of a “double compression”

In the photo world, there is a lot of JPEG images, and that involves some sort of compression. If we can detect double compression, that would mean that our images have been saved originally (compressed), and then manipulated and saved again (compressed again). This could be detected when there is a mismatch or desync of the blocks that we expect (8x8 in JPEG compression usually), and this could be used to detect copy and paste manipulation.

### 27 Feb 2020 (Digital Forensics)

#### We can exploit redundancy in data or limitations of people to improve compression

Since data has lots of redundancies as well as representation inefficiencies, we can look to these avenues in order to improve compression, or we can also improve compression based on the notion that people have limitations in visual processing capabilities, which makes compression acceptable even if we lose some quality.

#### Compression is composed of three parts:

1. Transformer (this performs one to one transformations of the data WITHOUT LOSS, in order to make things easier during compression (i.e: reduce correlation of data, etc)

2. Quantizer (this performs the mapping of many to one, and is where the loss comes from in lossy compression.)

3. Coder (assigns a code to each of the symbol produced by the quantizer, and is lossless as well, but represents data more efficiently, think huffman encoding)

You can achieve a lossless scheme by omitting the quantization, which is the lossy compression step, but of course performance in compression will degrade.

#### There are many useful properties of images to know to better compress

Block-based transformations split the image into blocks in order to be able to run things in parallel, as well as encode. This increases the efficiency.

Additionally, we can talk about colorspace compression, where we can go from RGB to YCbCr (Luminance Chrominance) Where the result is a Luma layer (essentially grayscale, and has most of the structure of the image) and the Chroma levels (Cr and Cb) and these contain the color information. We can subsample these, and take the luma for every pixel, but the chroma can be taken on a block in order to reduce the amount of information by half.

#### DCT – a transformation that assists in compression

The DCT transform can be likened to the FFT, except that we deal in real numbers, which makes our computations easier. We deal in the 2D. In the compression process, we apply DCT to the blocks of the image, which is nice because DCT has good energy compartment properties (high values are in the top left, meaning that when we apply quantization later we can preserve the data that keeps the most information regarding the image). We then apply the quantization (division of the Q matrix). Larger Q value means more compression.

#### Compression Footprints is detected via some artifacts

1. Blocking Artifacts (due to an blocking, images might sometimes have blocky overlaps, which can be detected)

2. Ringing Artifacts (due to sharp transitions, we could see a ghost or a shadow effect)

3. Graininess Artifacts (around edges, we could see grainy effects)

4. Blurry Artifacts (removing higher DCT coefficients increases the smoothness of an image, and this could be detected)

#### Double Compression is a good telltale sign – we can check the bins

We can look at the histogram, and notice that when we have a double quantization, the bins of the histogram (where the information of the data is), will become more compact and shoved into less bins, which leads to empty bins or less filled out bins, and the histogram is telling in that compression (Q mat applied) occurred more than once.

### 06 Mar 2020 (Digital Forensics)

#### Machine Learning/Data-driven models for forensics can be applied

This is a new field that is very popular. Up until now, we’ve been really looking at statistical models and structural models. However, it is important to know the benefits of data-driven models, since we’re not always sure about the probability models, in certain cases the statistical approaches we’ve looked at before would be obsolete.

We can consider different levels of Data-driven models: ranging from simple SVM (linear models) -> simple NN -> Deep net or Convolutional NN. For instance, a simple SVM linear model would be useful to solve the issue we had previously where when the Qfactors were similar, that it would be difficult to find good separation. With good training data, we could perform good separation even on those cases.

It is important to note that in the data-driven models we do have to be careful in terms of real life situations where binary classes are not balanced. For instance, when we’re training against counterfeit bank notes, it is expected that we will have a lot of training data on legit bank notes, but a lot less cases of fraudulent bank notes to test on.

#### Some pros and cons to the data-driven models

- no guarantee of solutions

- lots of data is required, and consumption of data is expensive

- not as consistent/universally applicable, especially with higher precision. Meaning that since we train on very specific cases to get good precise results, our models are all very specialized. It would be hard to take a fraud bank note checking model to then test for fraudulent medicinal packaging for instance.

+ performance is generally better

+ works well in good conditions for extremely hard to separate cases (ie: QF1 = QF2)