Photo Forensics from JPEG Rounding Artifacts

Image Processing and Security

Student:
Paula Mihalcea

Professor:
Alessandro Piva

August 26th, October 19th 2021

Photo forensics

- Photographs have always been **manipulated**
  - to **distort** or **change** their content, for a lot of reasons.
- Modern technology makes it easier than ever.
- **Photo forensics** is the science of **identifying** any **tampering**.

Original (left) and manipulated (right) version of a photo.

Coding-based forensics

- **Coding-based techniques** are among the most relevant forensic methods.

- They leverage **statistical correlations** introduced by lossy compression algorithms:
  - compression introduces distinct **artifacts,**
  - tampering creates **inconsistencies** in them;
    - ➢ both are normally invisible to the naked eye.

Image compression

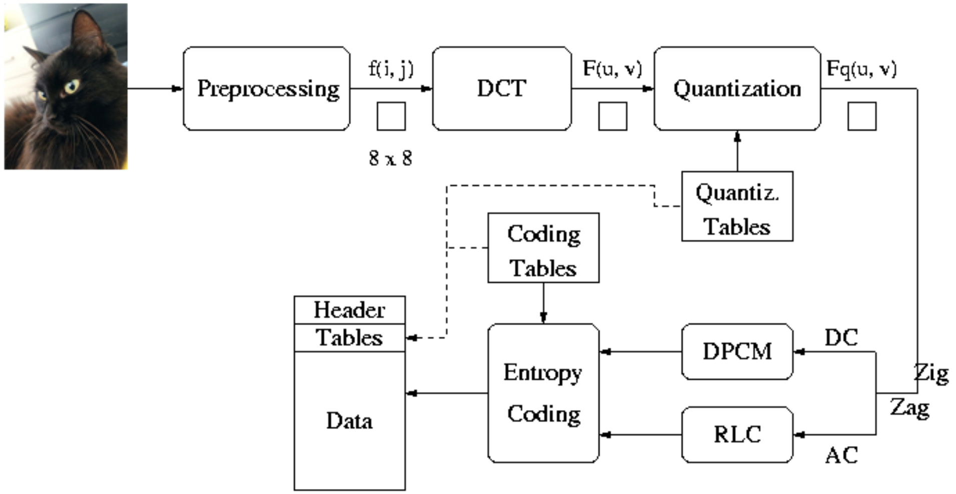
- **Reduces the data** representing a digital image, for:
  - efficient storage,
  - faster transmission.
- Information can be **encoded** with less data:
  - **lossless**: no information loss;
  - **lossy**: original content no longer retrievable.
- A popular compression method is the **JPEG standard**.
  - Introduced in 1992.
  - *Joint **P**hotographic **E**xperts **G**roup.*

Image compression

Three levels of JPEG compression.
From left to right: original image, medium compression, maximal compression.

JPEG compression

1. Color space conversion
2. Chroma subsampling
3. 8x8 px block partitioning
4. Level offset
5. **2D DCT conversion**
6. **Quantization**
7. Encoding



The JPEG compression algorithm.

2D DCT

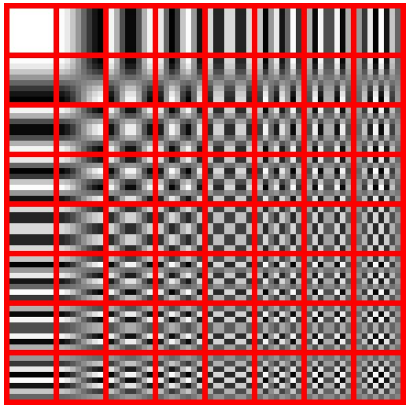
$$F(u,v)=\frac{C(u)C(v)}{4}\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\left(\frac{2(x+1)u\pi}{16}\right)\cos\left(\frac{2(y+1)v\pi}{16}\right)$$

$f(x,y)$: 2D sample value

$F(u,v)$: 2D DCT coefficient

$$C(x)=\begin{cases}\frac{1}{\sqrt{2}} & if\ x=0\\ 1 & otherwise\end{cases}$$

- The DCT transforms each block in a **linear combination** of 64 orthogonal basis signals.
  - Coefficients specifies the amount of each 2D spatial frequency of the image.

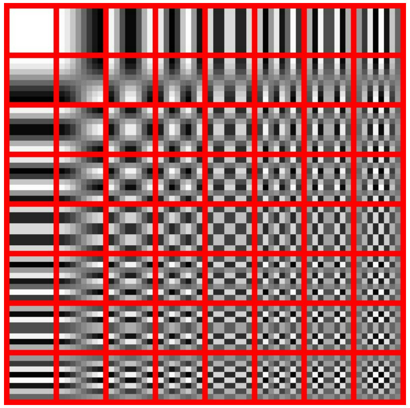


Quantization

- Coefficients at the bottom right of the matrix represent **higher frequencies** (details) and can be **discarded**.
  - The human eye does not notice the difference.
- During quantization each DCT coefficient $F(u,v)$ is **divided** by a constant $Q(u,v)$ taken from a predefined table Q:

$$F^Q(u,v) = Round\left(\frac{F(u,v)}{Q(u,v)}\right)$$

- **Rounding** the result to an **integer** achieves the detail reduction.
  - Higher frequencies are nullified.
  - This is a **lossy** operation.



Rounding operators

- A DCT coefficient c quantized by q can be rounded with:
  - $\boldsymbol{round}\left(\frac{c}{q}\right)$
  - $\boldsymbol{floor}\left(\frac{c}{q}\right) = \left\lfloor\frac{c}{q}\right\rfloor$
  - $\boldsymbol{ceiling}\left(\frac{c}{q}\right) = \left\lceil\frac{c}{q}\right\rceil$
- Every camera uses one of these operators.
- Agarwal & Farid [1] noted that the **floor/ceiling operators** consistently yield smaller/larger values in the top-left pixel of each 8x8 block.
  - This results in a **periodic artifact**.

Rounding example (1D)

Frequency domain

- $\vec{s} = (3.7\ 8.3\ 5.9)$ input 1D signal
- $\overrightarrow{\boldsymbol{s_f}} = (3 \quad 8 \quad 5) = \vec{s} + (-0.7\ -0.3\ -0.9) \approx \vec{\boldsymbol{s}} - \boldsymbol{\alpha_f}\vec{\mathbf{1}}$ quantized values (floor)
- $\overrightarrow{\boldsymbol{s_c}} = (4 \quad 9 \quad 6) = \vec{s} + (0.3 \qquad 0.7 \qquad 0.1) \approx \vec{\boldsymbol{s}} + \boldsymbol{\alpha_c}\vec{\mathbf{1}}$ quantized values (ceiling)

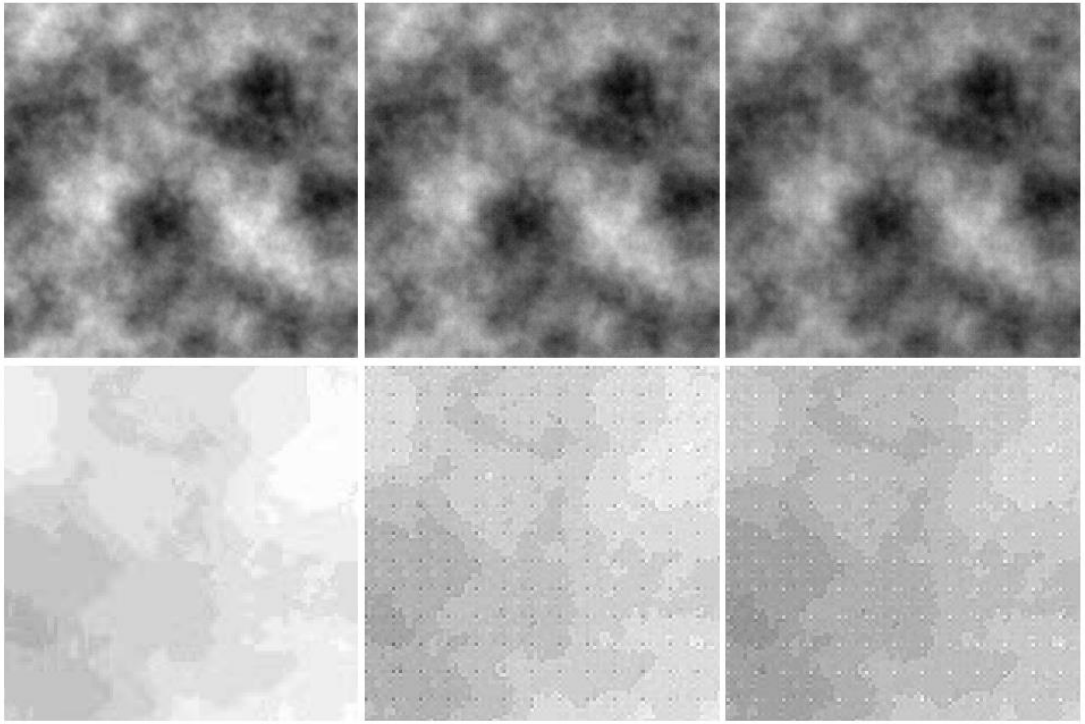
where $\alpha_{(\cdot)}$ is the mean of $\overrightarrow{s_{(.)}} - \vec{s}$ and $\vec{1} = (1 \ldots 1)$ is a constant signal.

Spatial domain (inverse DCT)

- $$\begin{aligned} F^{-1}\left(\overrightarrow{s_f}\right) &= F^{-1}\left(\vec{s} - \alpha_f\vec{1}\right) \\ &= F^{-1}(\vec{s}) - F^{-1}\left(\alpha_f\vec{1}\right) \\ &= F^{-1}(\vec{s}) - \alpha_f\vec{\delta} \end{aligned}$$
- $$\begin{aligned} F^{-1}\left(\overrightarrow{s_c}\right) &= F^{-1}\left(\vec{s} + \alpha_c\vec{1}\right) \\ &= F^{-1}(\vec{s}) + F^{-1}\left(\alpha_c\vec{1}\right) \\ &= F^{-1}(\vec{s}) + \alpha_c\vec{\delta} \end{aligned}$$

JPEG dimples

- Periodic artifacts consisting of a **single darker** (floor) or **brighter** (ceiling) **pixel** in the upper left corner of each 8x8 block.



JPEG images compressed with one of three rounding operators:
round (left), floor (center), ceiling (right).

The bottom row shows a magnified view of the upper left corner of each image.

The periodic JPEG dimple artifacts are introduced by the floor and ceiling operators, but not the round operator.

Dimple detection: preprocessing

1. **Preprocessing** *to suppress noise from the underlying image content.*
   a. Wiener filter (3x3) to each RGB channel.
   b. Average of residual noise across all channels.
   c. Average of non-overlapping 32x32 blocks across the image (or part of it).

   This step returns a single 32x32 average block, b.

Close-up of a cat's fur before (left) and after (right) steps *a* and *b*.

Dimple detection: PCE

2. PCE

$$p_I = \frac{F_I^2(\hat{u},\hat{v})}{\frac{1}{63}\sum_{(u,v)\neq(\hat{u},\hat{v})} F_I^2(u,v)}$$

peak-to-correlation energy for $\boldsymbol{I(x,y)}$

where $$F_I(u,v) = \sum_x \sum_y I(x,y)T(x+u,y+v),$$

2D cross-correlation between $I(x,y)$ and $T(x,y)$

assuming that:

- $I(x,y)$ original image;
- $T(x,y)$ 32x32 template of all 0s except a 1 in the upper left corner of every 8x8 block;
- $I(\cdot)$, $T(\cdot)$ zero-mean and unit-sum;
- $(u,v) \in [0,7]$ spatial offsets due to 8 pixels periodicity;
- $(\hat{u},\hat{v})$ offset that maximises $F_I(\cdot)$: $(\hat{u},\hat{v}) = \arg\max_{u,v}(F_I^2(u,v))$.

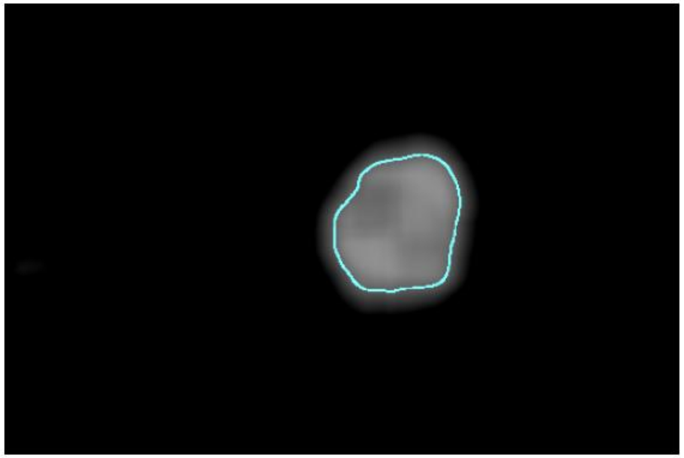
Dimple detection: map

3. Prominence map

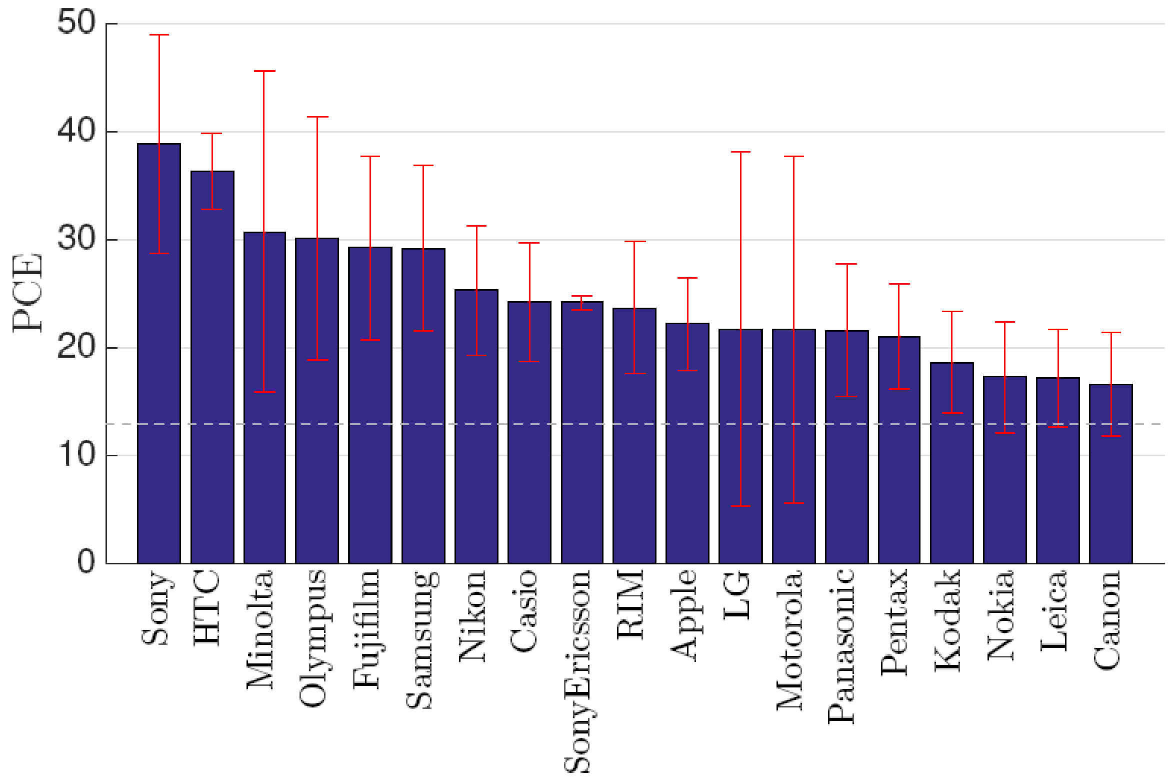
- PCE measures the dimple strength for the **entire image**:
  - it is computed against the 32x32 averaged block b and $T(x, y)$;
  - large PCE = more prominent dimples.
- To detect local manipulations, step 2 is applied instead to **overlapping 512x512 windows**, yielding a prominence map.

- The prominence map specifies the **per-pixel dimple strength**.
  - Each pixel contains the average PCE of all 512x512 windows containing it.

Dimple detection: results



Dimple detection: results



Average dimple strength per camera manufacturer.
Each bar corresponds to the average PCE value for all available models.
The dashed line is the detection threshold, empirically set at 13.

Dimples properties

- Dimples are normally produced by acquisition devices (not software).
- PCE responds regardless of where in the 8x8 block the impulse appears, so dimples can survive:
  - **cropping**;
  - dimples **grid offset**;
  - **90° rotations**;
  - **recompression** with JPEG quality ≥ 30;
  - **additive noise** ≤ -10 dB;
  - **scaling** ≤ 0.6;
  - format **conversion**.

Beyond the dimples

- We can model a **JPEG encoder** to capture more generic rounding-based artifacts [2].
- By estimating its parameters, we can:
  - understand the dimples' **variability**;
  - **reproduce rounding errors** observed in real-world cameras;
  - exploit these artifacts to **localize image manipulation**.
- Actually, only the **DCT conversion** needs to be modeled.
  - It is the most computationally demanding step.
  - Different implementations lead to diverse speed/accuracy tradeoffs.

A generic JPEG algorithm

- **Fixed parameters**
  - fixed-point arithmetic;
  - 32-bit precision;
  - two's-complement representation;
  - two 1D DCTs, instead of a 2D DCT.
- **Configurable parameters**
  - $d(\cdot)$ 1D DCT:
    - $d_1(\cdot)$ Ligtenberg-Vetterli;
    - $d_2(\cdot)$ Loeler-Ligtenberg-Moschytz;
    - $d_3(\cdot)$ Arai-Agui-Nakajima;
    - All implemented with additions and multiplications.
  - $\vec{s}_h$, $\vec{s}_v$ de-scaling 8D vectors;
  - s de-scaling scalar for the 2D DCT coefficients after two 1D DCTs;

block-jpeg(I)

$I = I - 128$ ▹ normalize into [-128,127]

▹ 2-D block DCT from two, 1-D DCTs

for all rows i in I **do** ▹ 1-D DCT on row

  $\mathcal{I}(i,:) = d(I(i,:)', \vec{s}_h, f_e(\cdot), f_o(\cdot))$

for all columns j in $\mathcal{I}$ **do** ▹ 1-D DCT on column

  $\mathcal{I}(:,j) = d(\mathcal{I}(:,j), \vec{s}_v, f_e(\cdot), f_o(\cdot))$

▹ Scale and quantize DCT coefficients

for all rows i in $\mathcal{I}$ **do**

  for all columns j in $\mathcal{I}$ **do**

    $\mathcal{I}(i,j) = f_s(\mathcal{I}(i,j), s)$ ▹ 2-D de-scale

    if $Q(i,j)$ is a power of 2 **then**

      $\mathcal{I}(i,j) = f_2(\mathcal{I}(i,j), Q(i,j))$ ▹ quantize

    else

      $\mathcal{I}(i,j) = f_q(\mathcal{I}(i,j), Q(i,j), 0)$ ▹ quantize

return $\mathcal{I}$

A generic JPEG algorithm

- **Configurable parameters**
  - Q 8x8 quantization matrix;
  - $f_e(\cdot)$, $f_o(\cdot)$ de-scaling rounding operators for even and odd DCT coefficients;
  - $f_s(\cdot)$ 2D DCT de-scaling rounding operator;
  - $f_q(\cdot)$ final quantization rounding operator for non power-of-two values;
  - $f_2(\cdot)$ final quantization rounding operator for power-of-two values.

---

block-jpeg(I)

$I = I - 128$ ▷ normalize into [-128,127]

▷ 2-D block DCT from two, 1-D DCTs

for all rows i in I **do** ▷ 1-D DCT on row

  $\mathcal{I}(i,:) = d(I(i,:)', \vec{s}_h, f_e(\cdot), f_o(\cdot))$

for all columns j in $\mathcal{I}$ **do** ▷ 1-D DCT on column

  $\mathcal{I}(:,j) = d(\mathcal{I}(:,j), \vec{s}_v, f_e(\cdot), f_o(\cdot))$

▷ Scale and quantize DCT coeffficients

for all rows i in $\mathcal{I}$ **do**

  for all columns j in $\mathcal{I}$ **do**

    $\mathcal{I}(i,j) = f_s(\mathcal{I}(i,j), s)$ ▷ 2-D de-scale

    if $Q(i,j)$ is a power of 2 **then**

      $\mathcal{I}(i,j) = f_2(\mathcal{I}(i,j), Q(i,j))$ ▷ quantize

    else

      $\mathcal{I}(i,j) = f_q(\mathcal{I}(i,j), Q(i,j), 0)$ ▷ quantize

return $\mathcal{I}$

Rounding operators

- $f_e(\cdot)$, $f_o(\cdot)$, $f_s(\cdot)$, $f_q(\cdot)$, $f_2(\cdot)$ can be implemented as:
  - **round**: rounds to the nearest integer;
    $round(1{,}5) = 2 \qquad round(-1{,}5) = -2$
  - **halfup**: like round, but in ties rounds to the largest nearest integer;
    $halfup(1{,}5) = 2 \qquad halfup(-1{,}5) = -1$
  - **trunc**: rounds to the smallest nearest integer ($\rightarrow 0$);
    $trunc(1{,}5) = 1 \qquad trunc(-1{,}5) = -1$
  - **floor**: rounds to the smallest nearest integer ($\rightarrow -\infty$).
    $floor(1{,}5) = 1 \qquad floor(-1{,}5) = -2$

Rounding operators bias

- Operators have a **bias** and **speed/accuracy tradeoff**.
- Consider the case of dividing by a power of 2:
  - Δ_r, Δ_h, Δ_t and Δ_f random variables s.t. $\Delta_* = -\frac{x}{2^k} + f_*(\frac{x}{2^k})$, and:
    - uniformly distributed in a finite interval,
    - uncorrelated with the input x,
    - independent, at any step, from any other step in the algorithm.
- For example, take Δ_f:
  - for $k = 1$ its expected value is $-0{,}25$;
  - for large k, this values approaches $0{,}5$;
  - conclusion: the floor operator introduces a consistent negative bias.

Rounding operators bias

- Proof:
  - $\Delta_f = -\frac{x}{2^k} + floor\left(\frac{x}{2^k}\right)$ uniformly distributed over $\frac{i}{2^k}$, $\mathrm{i} \in [-2^k + 1, 0]$
  - The **expected value** of the random variable Δ_f is:

$$\begin{aligned}
\mathbb{E}\left[\Delta_f\right] &= \frac{1}{2^k} \sum_{i=-(2^k-1)}^{0} \frac{i}{2^k} \\
&= \frac{1}{2^{2k}} \sum_{i=-(2^k-1)}^{0} i = -\frac{1}{2^{2k}} \sum_{i=0}^{2^k-1} i \\
&= -\frac{1}{2^{2k}} \left(\frac{2^k(2^k-1)}{2}\right) = -\frac{(2^k-1)}{2^{k+1}}.
\end{aligned}$$

Rounding operators bias

- The same happens for Δ_h, with opposite sign.
- On the contrary, **round and trunc do not introduce a consistent bias**.
- Coefficients of natural images are zero-mean,
  while **biases lead to non zero-mean distributions**.
- The result is a series of specific artifacts that vary across camera manufacturers, depending on:
  - rounding operator chosen;
  - de-scaling factors;
  - quantization factors.

JPEG parameter estimation

- X **expected value of the rounding artifact**
  - estimated by averaging all 8x8 DCT block from images compressed with a fixed JPEG encoder and quality;
- $\vec{\theta} = [d(\cdot), \vec{s}_h, \vec{s}_v, s, Q, f_q(\cdot), f_2(\cdot), f_e(\cdot), f_o(\cdot), f_s(\cdot)]$

  block-jpeg **parameters** yielding an artifact $Y_{\vec{\theta}}$ minimizing L_1 error:

$$\vec{\theta}_m = \arg\min_{\vec{\theta}} \left[\frac{1}{64} \sum_{i=0}^{7} \sum_{j=0}^{7} \left| X_{i,j} - Y_{\vec{\theta},i,j} \right| \right]$$

  where **estimated rounding artifact** $Y_{\vec{\theta}}$ is obtained by:
  - compressing a fixed set of 10 grayscale images[1] with *block-jpeg*;
  - averaging all non-overlapping 8x8 DCT blocks;
  - subtracting the floating-point average DCT block[2].

[1]10 512x512 images, for a total of 4.096·10 8x8 blocks.
[2]Obtained with floating-point DCT, to not introduce any rounding bias.

JPEG parameter estimation

- $\vec{\theta}_m$ is estimated with a **brute force search**.
- Full search space for configurable parameters is enormous:

$$\vec{\theta} = [d(\cdot), \vec{s}_h, \vec{s}_v, s, Q, f_q(\cdot), f_2(\cdot), f_e(\cdot), f_o(\cdot), f_s(\cdot)]$$

- We search over a **reduced search space**:
  - $\vec{s}_h, \vec{s}_v, s$ are restricted to powers of 2;
  - $\vec{s}_h, \vec{s}_v$ chosen s.t. coefficients after each 1D DCT have a maximum scale of 2^3;
  - $\vec{s}_h$ has the same value at all even/odd positions;
  - Q is known and fixed.

JPEG parameter estimation: software artifacts

$$\vec{\theta} = [d(\cdot), \vec{s}_h, \vec{s}_v, s, Q, f_q(\cdot), f_2(\cdot), f_e(\cdot), f_o(\cdot), f_s(\cdot)]$$

- **JPEGLib[1]/MATLAB/TurboJPEG ground truth parameters**:
  - $d = d_2$
  - $(\vec{s}_h^e, \vec{s}_h^o) = (2^{11}, 2^{11})$
  - $(\vec{s}_v^e, \vec{s}_v^o) = (2^{15}, 2^{15})$
  - $s = 1$ *no 2D de-scaling*
  - Q s.t. contains both powers and non-powers of 2
  - $f_q(\cdot) = f_2(\cdot) = round(\cdot)$
  - $f_e(\cdot) = f_o(\cdot) = halfup(\cdot)$
  - $f_s(\cdot) = *$ *divisor is 1, so all operators give the same L_1 error*
- $Y_{\vec{\theta}}$ constructed as explained before from the estimated $\vec{\theta}$.
- X computed using the same images of $Y_{\vec{\theta}}$.

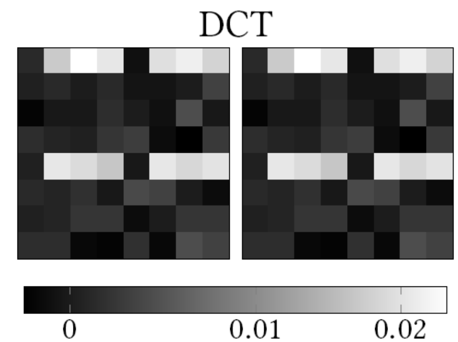
[1] Referring to the slow-integer DCT variant, a 13-bit precision 1D DCT implementation.

JPEG parameter estimation: software artifacts

d	(s_h^e, s_h^o)	(s_v^e, s_v^o)	s	$f_2(\cdot)$	$f_q(\cdot)$	$f_s(\cdot)$	$f_e(\cdot)$	$f_o(\cdot)$
d_2	$(2^{11}, 2^{11})$	$(2^{15}, 2^{15})$	2^0	round	round	*	halfup	halfup
d_2	$(2^{11}, 2^{11})$	$(2^{15}, 2^{15})$	2^0	round	round	round	halfup	halfup
d_2	$(2^{11}, 2^{11})$	$(2^{15}, 2^{15})$	2^0	round	round	halfup	halfup	halfup
d_2	$(2^{11}, 2^{11})$	$(2^{15}, 2^{15})$	2^0	round	round	trunc	halfup	halfup
d_2	$(2^{11}, 2^{11})$	$(2^{15}, 2^{15})$	2^0	round	round	floor	halfup	halfup
d_2	$(2^{11}, 2^{11})$	$(2^{15}, 2^{15})$	2^2	round	round	trunc	halfup	halfup
d_2	$(2^{11}, 2^{11})$	$(2^{15}, 2^{15})$	2^1	round	round	trunc	halfup	halfup

- Minimum L_1 error:
  10^{-17}
- As expected, $f_s(\cdot)$ introduces no artifacts.
- Parameters are **not** necessarily **unique**.

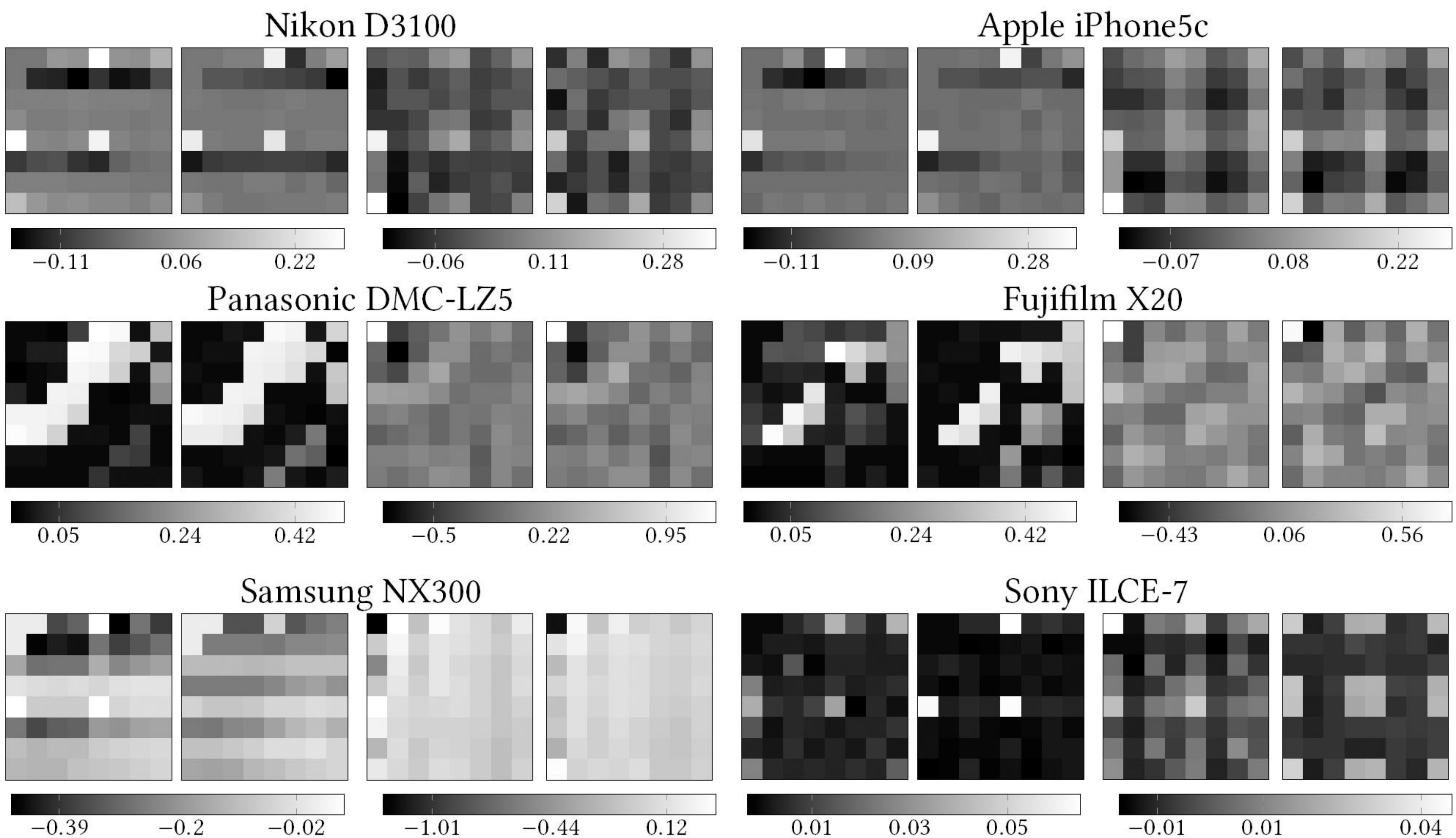
- Left: average 8x8 DCT blocks from JPEGLib images (X)
- Right: average 8x8 DCT blocks from *block-jpeg*($\vec{\theta}_m$) images ($Y_{\vec{\theta}_m}$)



JPEG parameter estimation: camera artifacts

- 24 different cameras
  - 25 to 100 images for each, with:
    - intact metadata;
    - same orientation;
    - same quality.
- Total: 1.694 images.
- Artifacts vary across manufacturers (structure and magnitude).
- Conclusion: different encoders introduce difference artifacts due to differences in JPEG compression parameters.
- **Note: some artifacts are not captured by the model.**

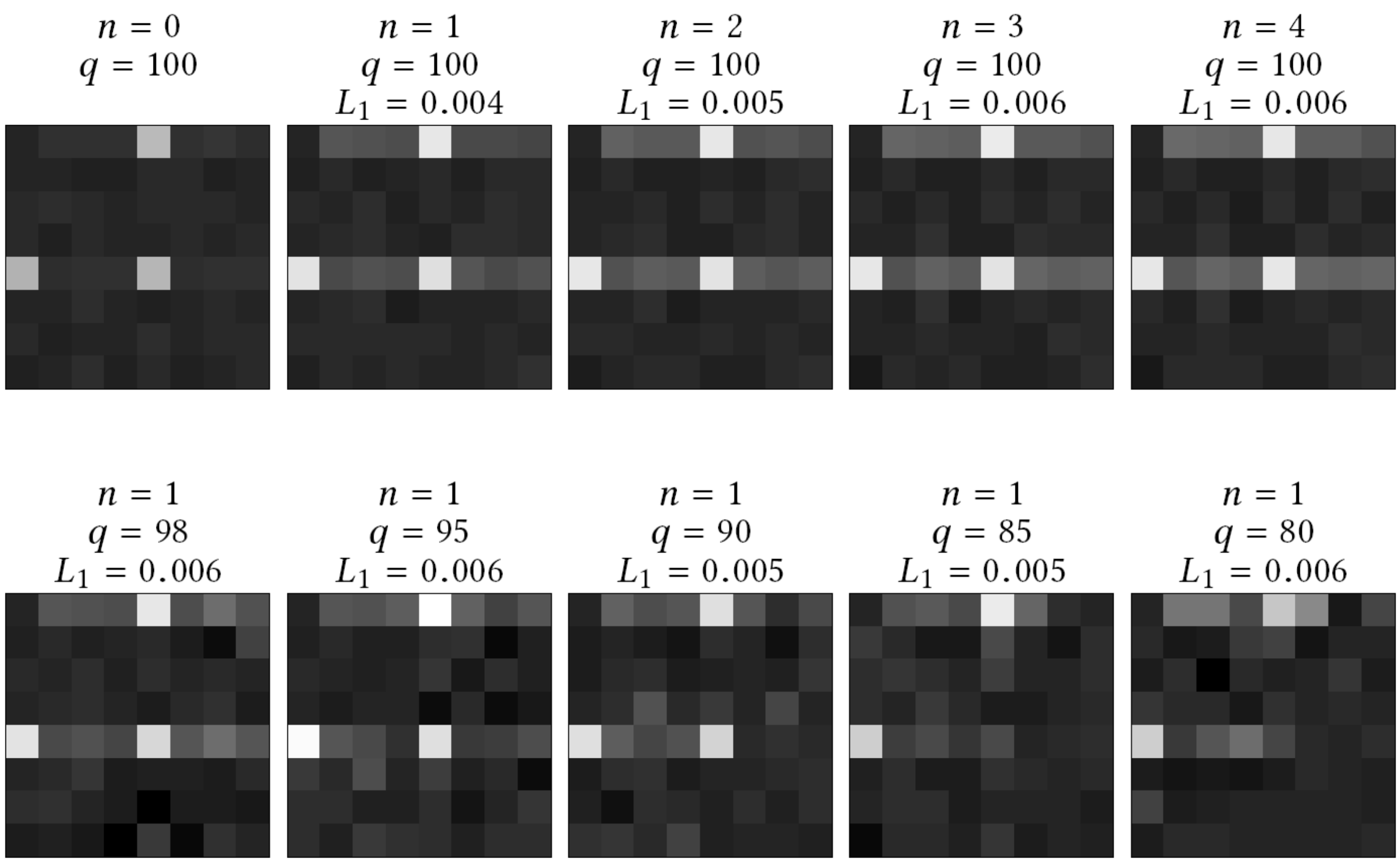
JPEG parameter estimation: camera artifacts



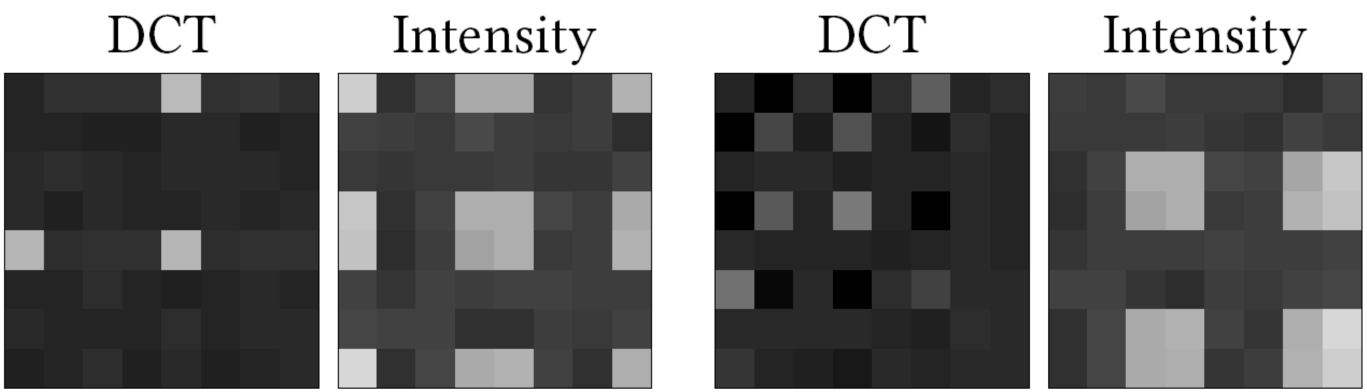
Post-manipulation artifacts

- Manipulating a JPEG image requires at least two compressions:
  - the **on-device compression** of the raw image;
  - the **software compression** of the photo editor used.
- They will likely have different encoders and quality.
- We shall consider the effects of:
  - **multiple compressions** on the device-induced artifacts;
  - misalignment of the original 8x8 lattice created by **cropping**.
- The following example configuration has 10 images created with the Sony ILCE-7 camera.

Post-manipulation artifacts: multiple compressions



Post-manipulation artifacts: cropping



Original artifact in the DCT and intensity domains.

Same artifact after cropping and saving in PNG lossless format.

Forensics

- Manipulated images will contain either:
  - a combination of artifacts (high-quality recompression),
  - diminished artifacts (low-quality recompression), or
  - new artifacts (cropping + recompression).
- Manipulated portions will have **artifacts inconsistent with the rest** of the image.
- We do *not* know the original compression parameters.
  - There is no practical way to consider all possible initial parameters and cropping offsets.
- Using a **data-driven approach**, we can automatically **estimate and segment** an image based on its different artifacts.

Expectation-Maximization algorithm

- This problem can be formulated within **the Expectation-Maximization (EM)** framework:
  - each 8x8 pixel block is assumed to belong to one of two classes:
    - C_1 original image;
    - C_2 manipulated portion, with unknown (possibly non-existent) rounding artifacts.
- Each EM iteration consists of:
  - estimating the rounding artifacts in each class as an 8x8 template in the intensity domain;
  - computing probability that each 8x8 block belongs to class C_1.
- Before proceeding, some preprocessing has to be done.

Preprocessing

1. RGB to YCbCr conversion and Cb/Cr channels removal
   - Rounding artifacts are highly correlated across color channels, so **only the luminance channel (Y) is analyzed**.

2. **3x3 median filtering**
   - Better at filtering salt-pepper noise, so more suitable for artifacts manifesting as a single darker/lighter pixel; reduces interference of image content. We **use the residual** of the filter.

Preprocessing

3. Luminance channel tiling

by 8 pixels, into overlapping **square windows** $\boldsymbol{w_i(x,y)}$ of size 64, 128 or 256.

4. $\vec{\boldsymbol{b}}_{\boldsymbol{i}}$ **computation**, $\forall\, w_i(x,y)$
   - It is the average block of all non-overlapping 8x8 blocks in window $w_i(x,y)$.

5. Random initialization of **template** $\vec{\boldsymbol{c}}_{\boldsymbol{1}}$ (uniform distribution in $[0,1]$).
   - It consists of the 64 intensity values of the rounding artifacts.
   - Class C_2 is not parametrized, and is treated as an outlier class;

   this way if there are more than two artifacts, one will fall under C_1 and all the others under C_2.

Expectation-Maximization: E

- **Conditional probability estimation** that each block $\vec{b}_i$ belongs to one of the two classes (C_1 and C_2), $\forall\ \vec{b}_i$:
- $P(\vec{b}_i \in C_1 | r_i) =$

$$= \frac{P(r_i | \vec{b}_i \in C_1) P(\vec{b}_i \in C_1)}{P(r_i | \vec{b}_i \in C_1) P(\vec{b}_i \in C_1) + P(r_i | \vec{b}_i \in C_2) P(\vec{b}_i \in C_2)}$$

where $r_i = \vec{b}_i \otimes \vec{c}_1$ is the correlation between $\vec{b}_i$ and template $\vec{c}_1$; we will shortly see how the other values are calculated.

- $P(\vec{b}_i \in C_2 | r_i) = 1 - P\left(\vec{b}_i \in C_1 \middle| r_i\right)$

numerically computed offline from $P\left(\vec{b}_i \in C_1 \middle| r_i\right)$.

Expectation-Maximization: E

$$P(\vec{b}_i \in C_1 | r_i) = \frac{P(r_i | \vec{b}_i \in C_1) P(\vec{b}_i \in C_1)}{P(r_i | \vec{b}_i \in C_1) P(\vec{b}_i \in C_1) + P(r_i | \vec{b}_i \in C_2) P(\vec{b}_i \in C_2)}$$

- $P(\vec{b}_i \in C_1) = 0{,}5$ prior assumption
- $P(\vec{b}_i \in C_2) = 0{,}5$ prior assumption
- $P(r_i | \vec{b}_i \in C_1)$:
  - the luminance of 1000 raw images is extracted and JPEG compressed with 18 random configurations previously estimated for camera artifacts, for a total of 18.000 images;
  - a single NxN window aligned to the 8x8 JPEG-block lattice is extracted from each image;
  - non-overlapping 8x8 pixel blocks are averaged for an estimate of the rounding artifacts;
  - an estimate of $P(r_i | \vec{b}_i \in C_1)$ is then yielded by correlating the blocks and their matching templates.
- $P(r_i | \vec{b}_i \in C_2)$:
  - like before, but using floating-point DCT during the JPEG compression (no rounding bias);
  - the final blocks are then correlated with the same camera templates.

Expectation-Maximization: M

- $\vec{c}_1$ **re-estimation** with a weighted average of all blocks $\vec{b}_i$:

$$\vec{c}_1 = \frac{\sum_i P(\vec{b}_i \in C_1 | r_i)\vec{b}_i}{\sum_i P(\vec{b}_i \in C_1 | r_i)}$$

the weight is the probability that each block $\vec{b}_i$ belongs to model C_1.

- The **E and M steps are performed iteratively**.
- The algorithm stops when the **difference between successive estimates of $\vec{c}_1$ is below a specified threshold**.

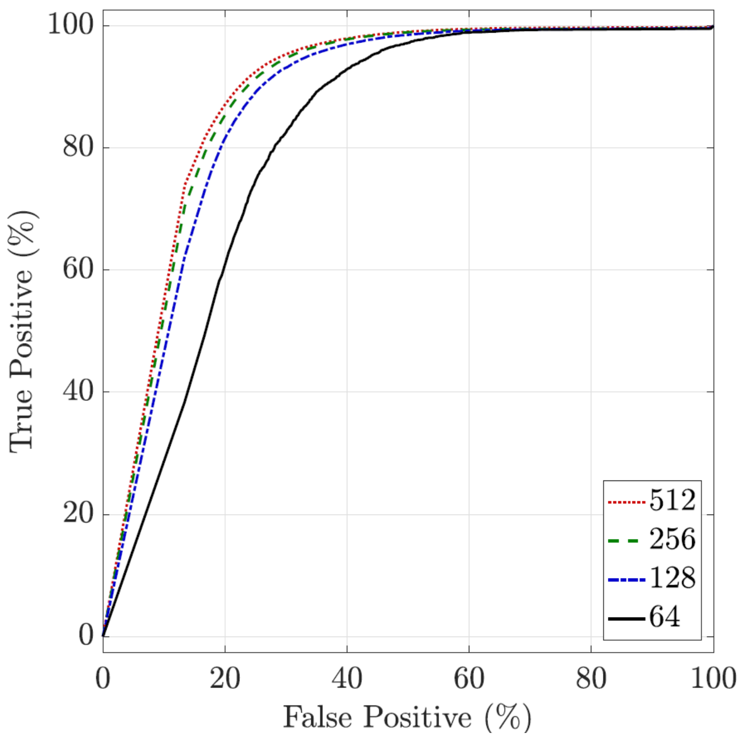
Experimental setup: paper

- **580 images** from 24 different cameras.
- **4 manipulations** to a square region of each image:
  - copy-move: the region duplicated;
  - median filter: a 3x3 median filter is applied to the region;
  - rotation: the region is randomly rotated (from 10° to 80°);
  - content-aware fill: the region is removed with a standard content-aware fill algorithm.
- Regions randomly varied in:
  - position (manipulations not always aligned to the 8x8 JPEG lattice);
  - size (64/128/256/512 px).
- Each image was analyzed with the EM algorithm.
  - Variable window $w(x, y)$ size (64, 128 and 256 pixels).
  - 580 images x 4 manipulations x 4 manipulation sizes = 9.280 images.

Experimental setup: Python implementation

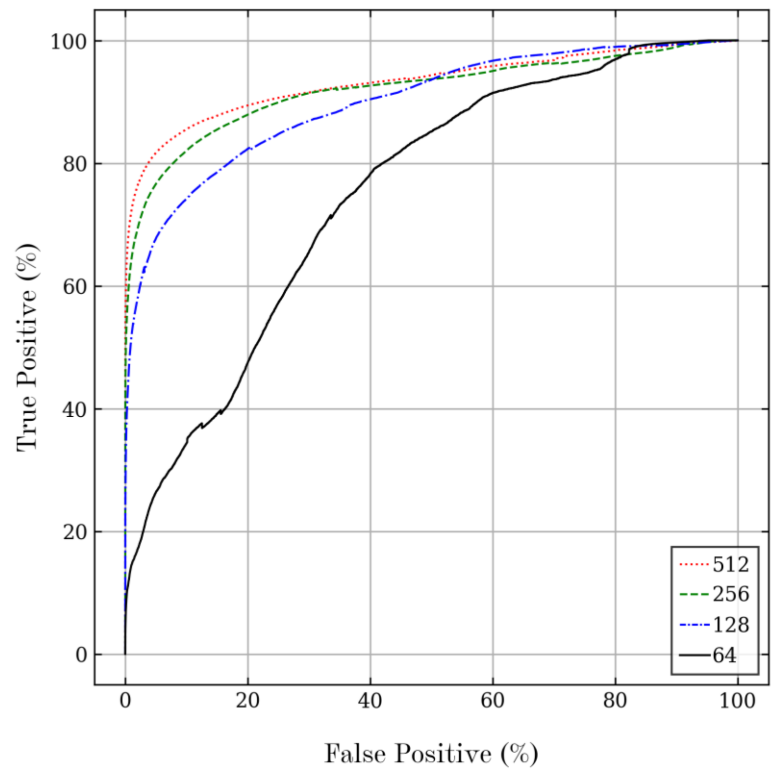
- 15 images from unknown cameras with variable dimples strength.
- 4 manipulations to a random square region of each image.
- 5 different formats:
  - PNG;
  - 4 JPEG compression qualities (60-70, 71-80, 81-90, 91-100).
- Each image was analyzed with the EM algorithm.
  - Variable window $w(x, y)$ size (64, 128 and 256 pixels).
- 15 images x 4 manipulations x 4 manipulation sizes x 5 formats = **1200 images**.
- 1200 images x 3 window sizes = **3600 analyses**.

Results: ROC curve



AUC score: 0,90 (512), 0,89 (256), 0,87 (128), 0,81 (64).

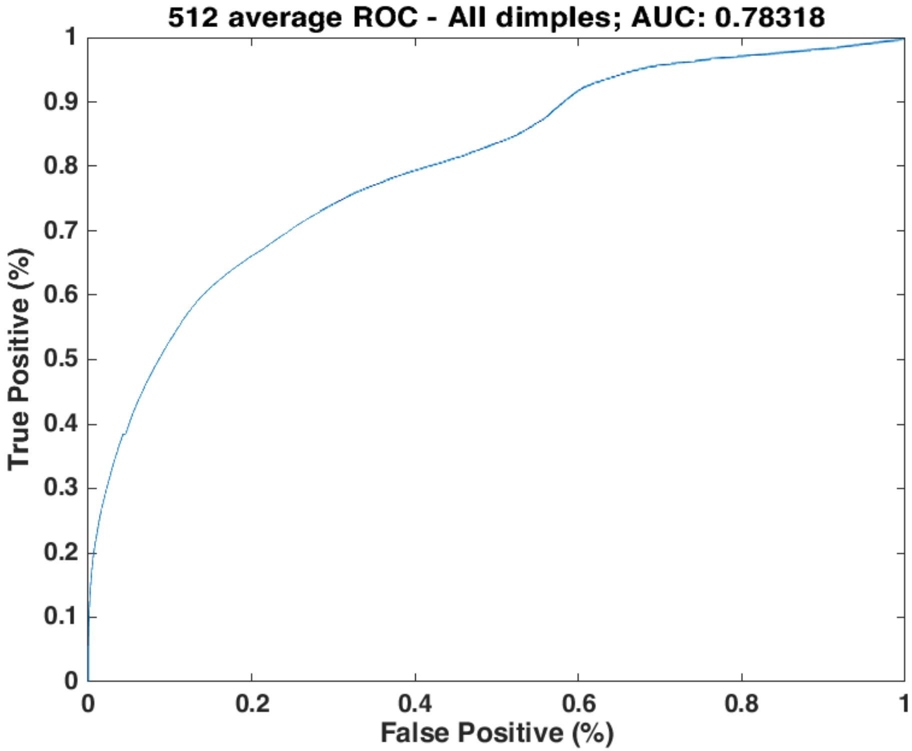
Paper average ROC.



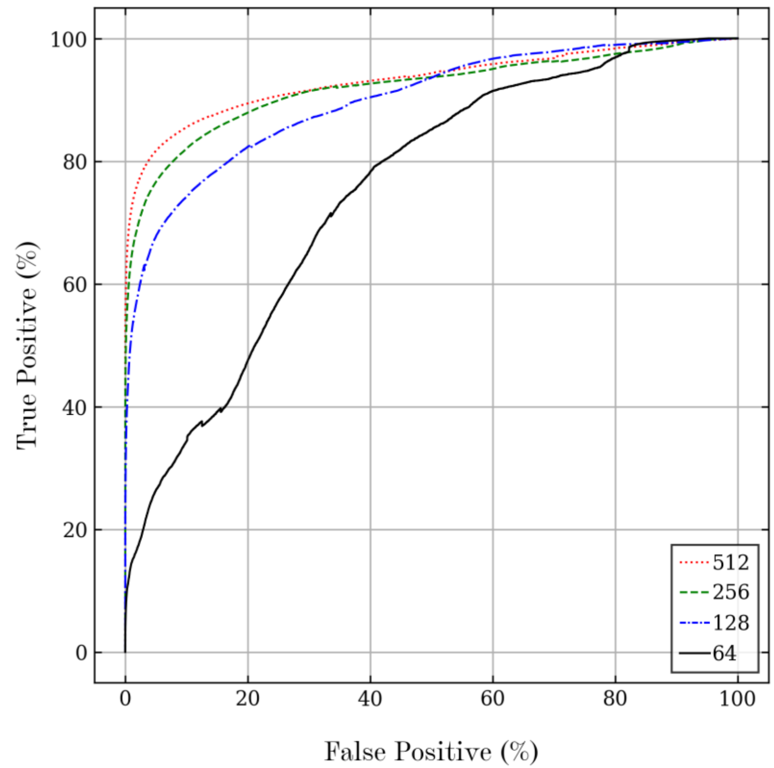
AUC score: 0.92 (512) , 0.90 (256), 0.85 (128), 0.73 (64)

Python implementation average ROC.

Results: ROC curve (paper 1)



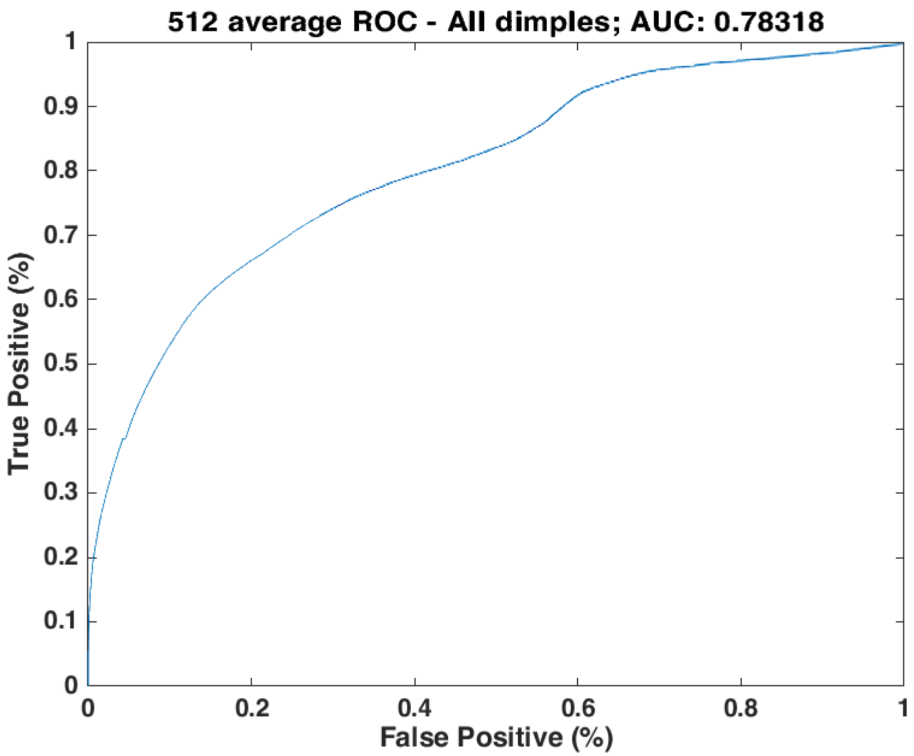
Paper 1 average ROC.
AUC score: 0,78 (512).

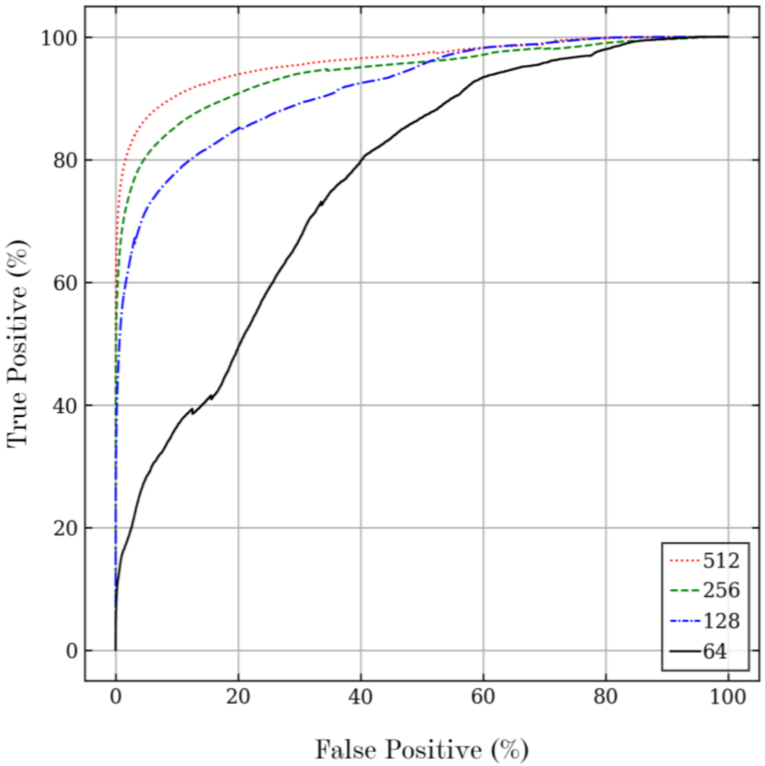


AUC score: 0.92 (512) , 0.90 (256), 0.85 (128), 0.73 (64)

Python implementation average ROC.

Results: ROC curve (without failure cases)



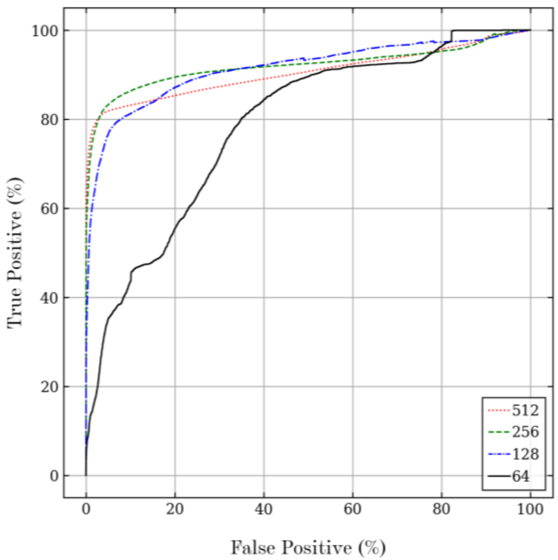


AUC score: 0.95 (512) , 0.92 (256), 0.87 (128), 0.74 (64)

Paper 1 average ROC.
AUC score: 0,78 (512).

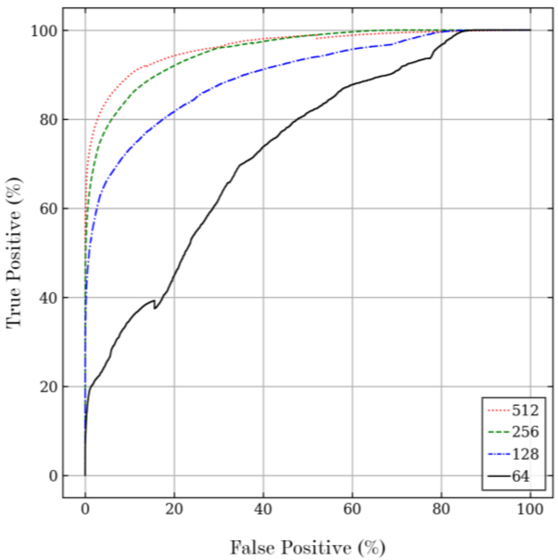
Python implementation average ROC.

ROC curve by dimples



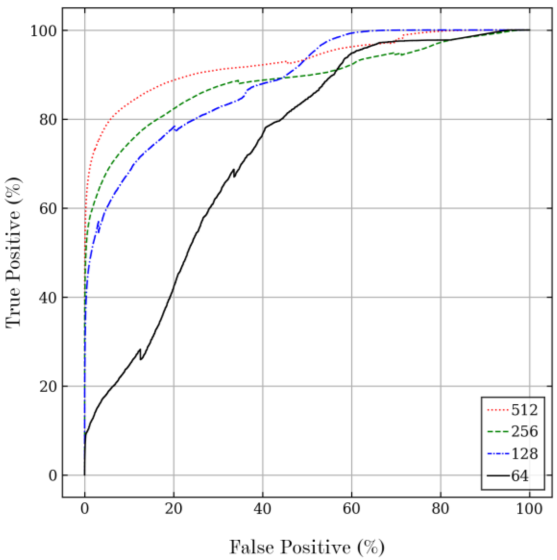
AUC score: 0.89 (512) , 0.90 (256), 0.86 (128), 0.73 (64)

Average ROC for high-strength dimples.



AUC score: 0.94 (512) , 0.93 (256), 0.85 (128), 0.71 (64)

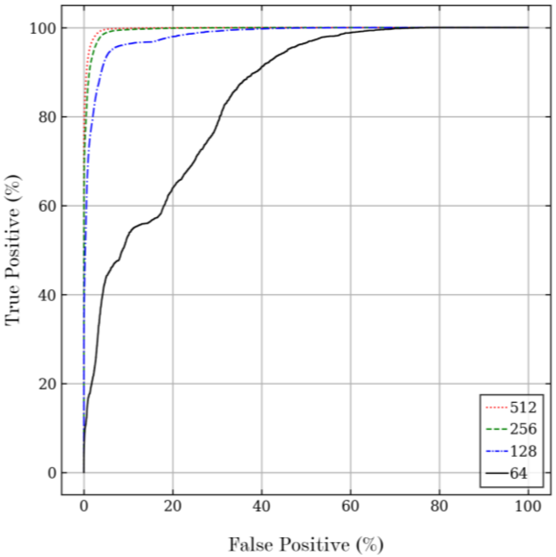
Average ROC for medium-strength dimples.



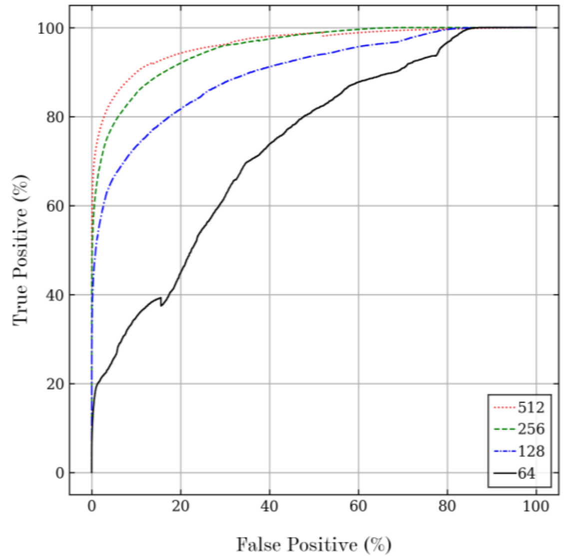
AUC score: 0.92 (512) , 0.87 (256), 0.84 (128), 0.73 (64)

Average ROC for low-strength dimples.

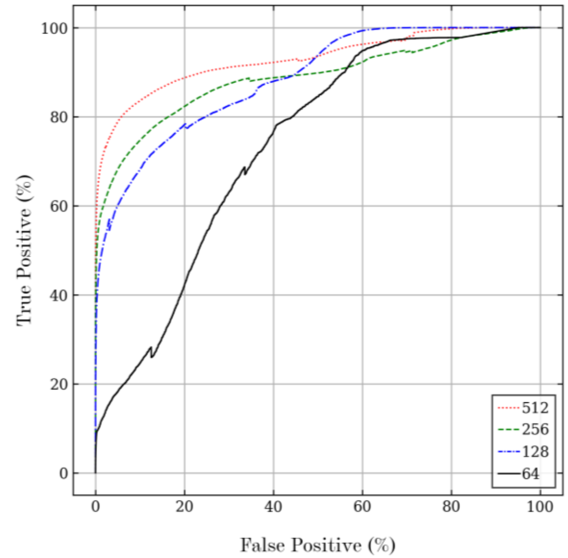
ROC curve by dimples (without failure cases)



AUC score: 0.99 (512) , 0.99 (256), 0.92 (128), 0.80 (64)



AUC score: 0.94 (512) , 0.93 (256), 0.85 (128), 0.71 (64)



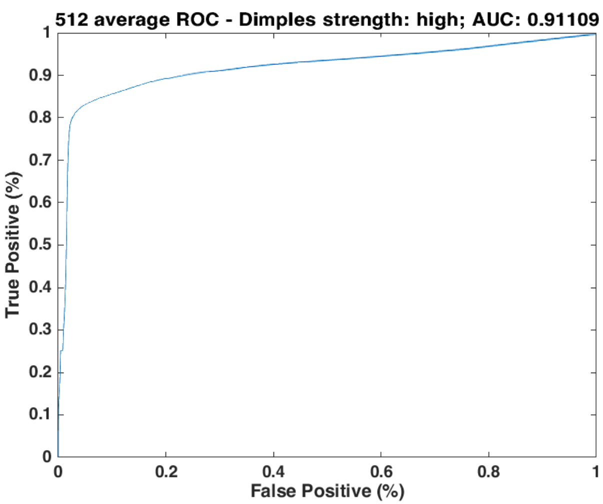
AUC score: 0.92 (512) , 0.87 (256), 0.84 (128), 0.73 (64)

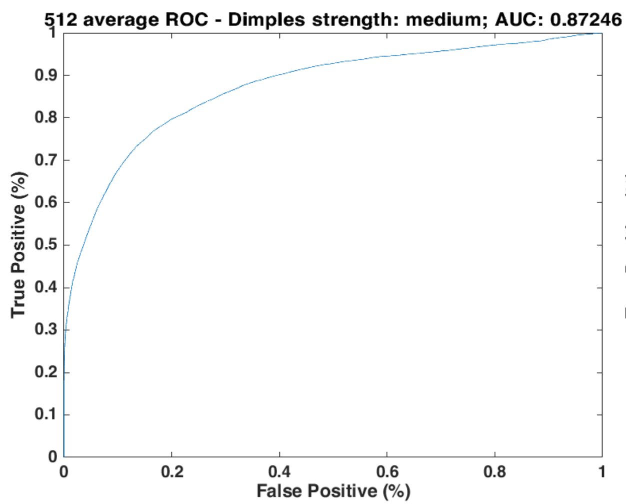
Average ROC for high-strength dimples.

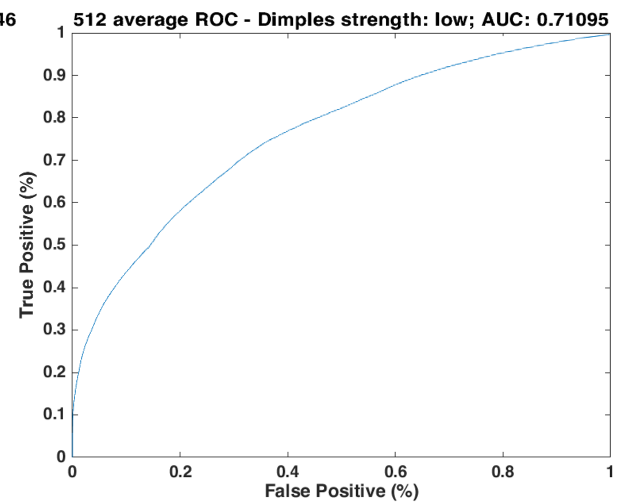
Average ROC for medium-strength dimples.

Average ROC for low-strength dimples.

ROC curve by dimples (paper 1)





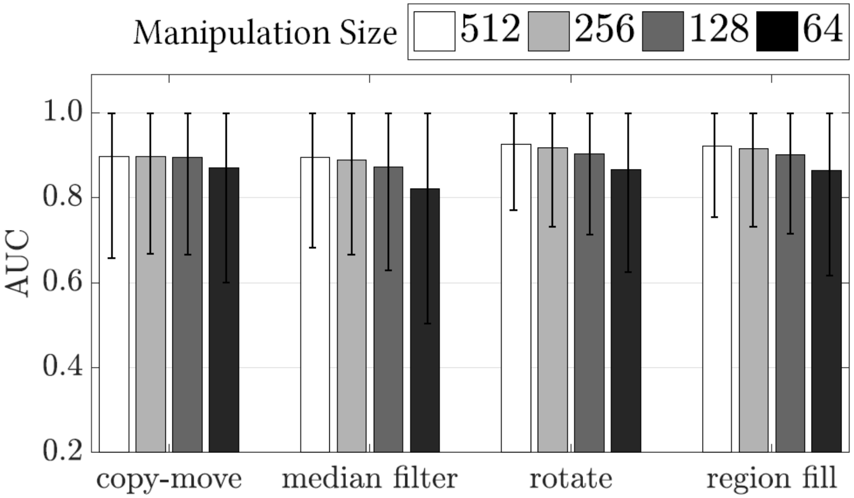


Average ROC for high-strength dimples.
AUC score: 0,91 (512).

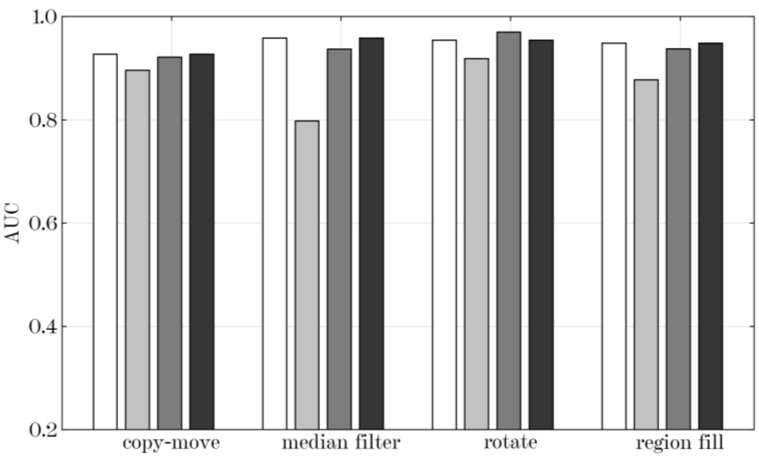
Average ROC for medium-strength dimples.
AUC score: 0,87 (512).

Average ROC for low-strength dimples.
AUC score: 0,71 (512).

AUC: manipulation type

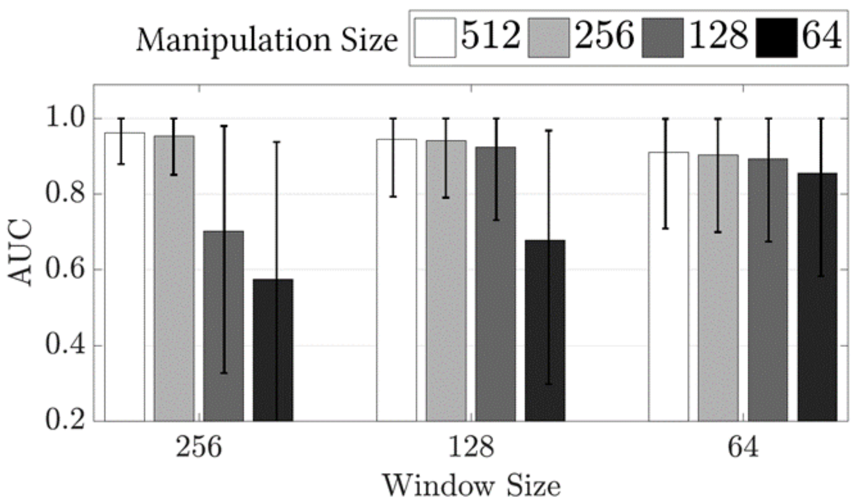


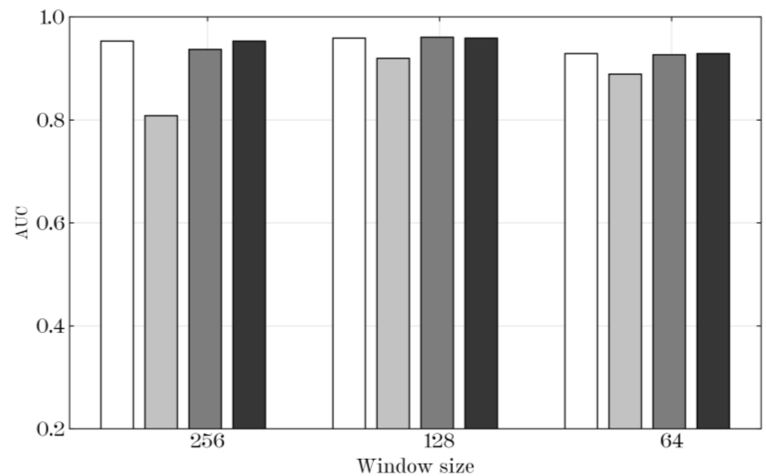
Paper AUC by manipulation type.



Python implementation AUC by manipulation type

AUC: window size

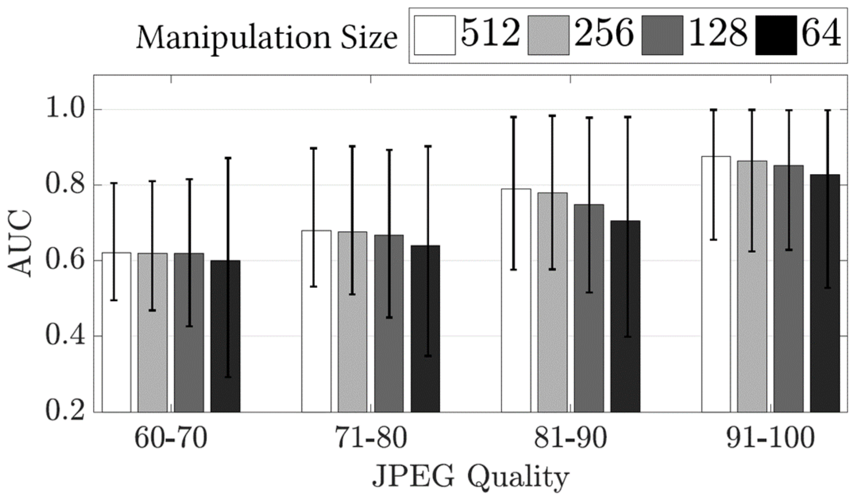




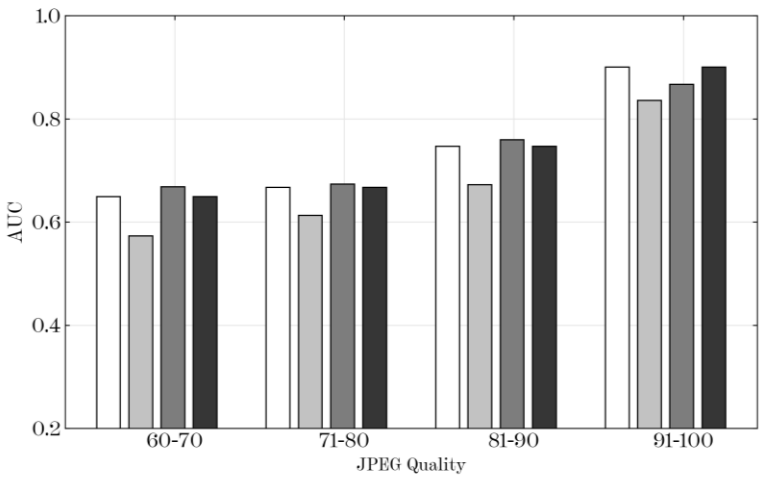
Paper AUC by EM window size.

Python implementation AUC by EM window size.

AUC: JPEG quality

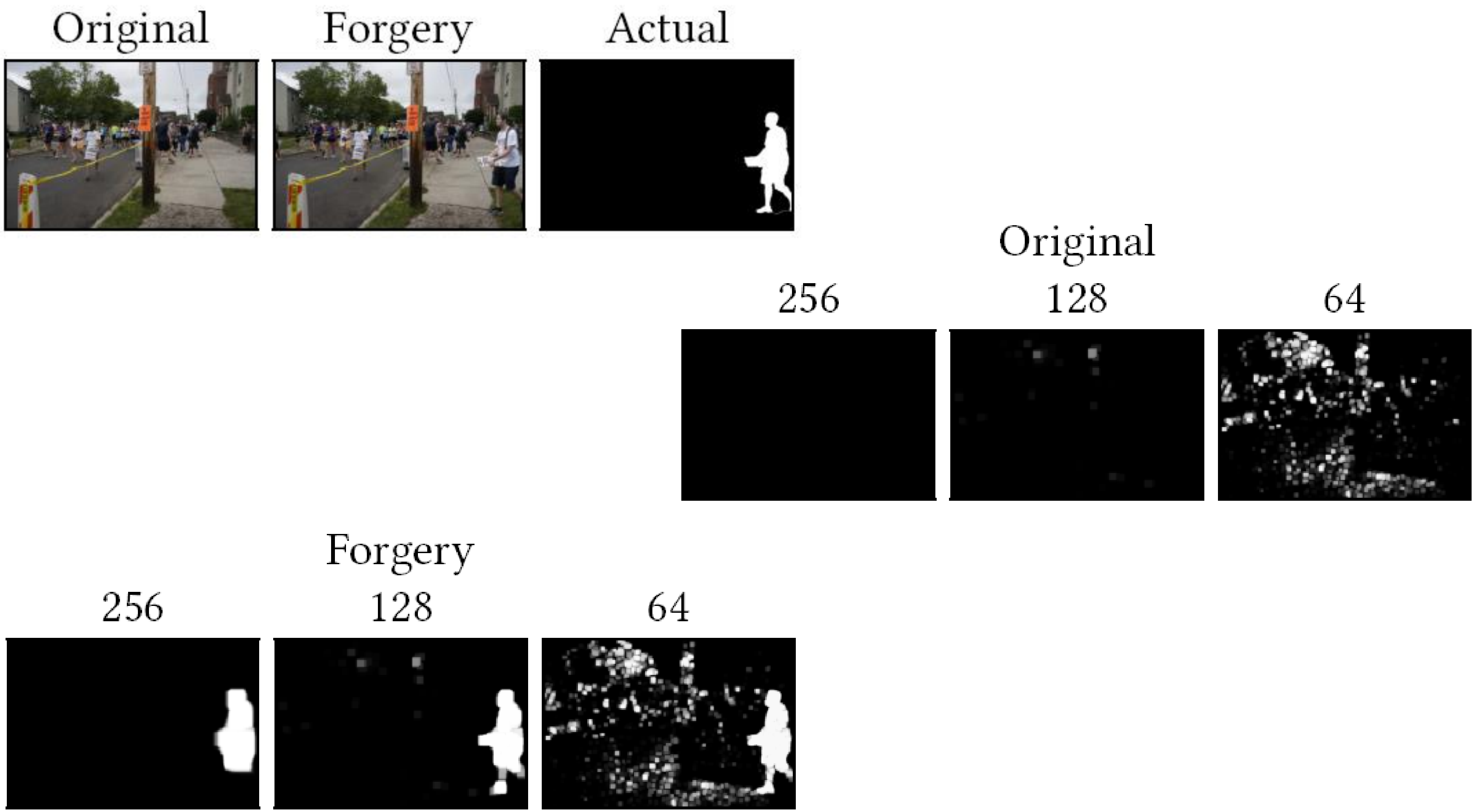


Paper AUC by JPEG quality.

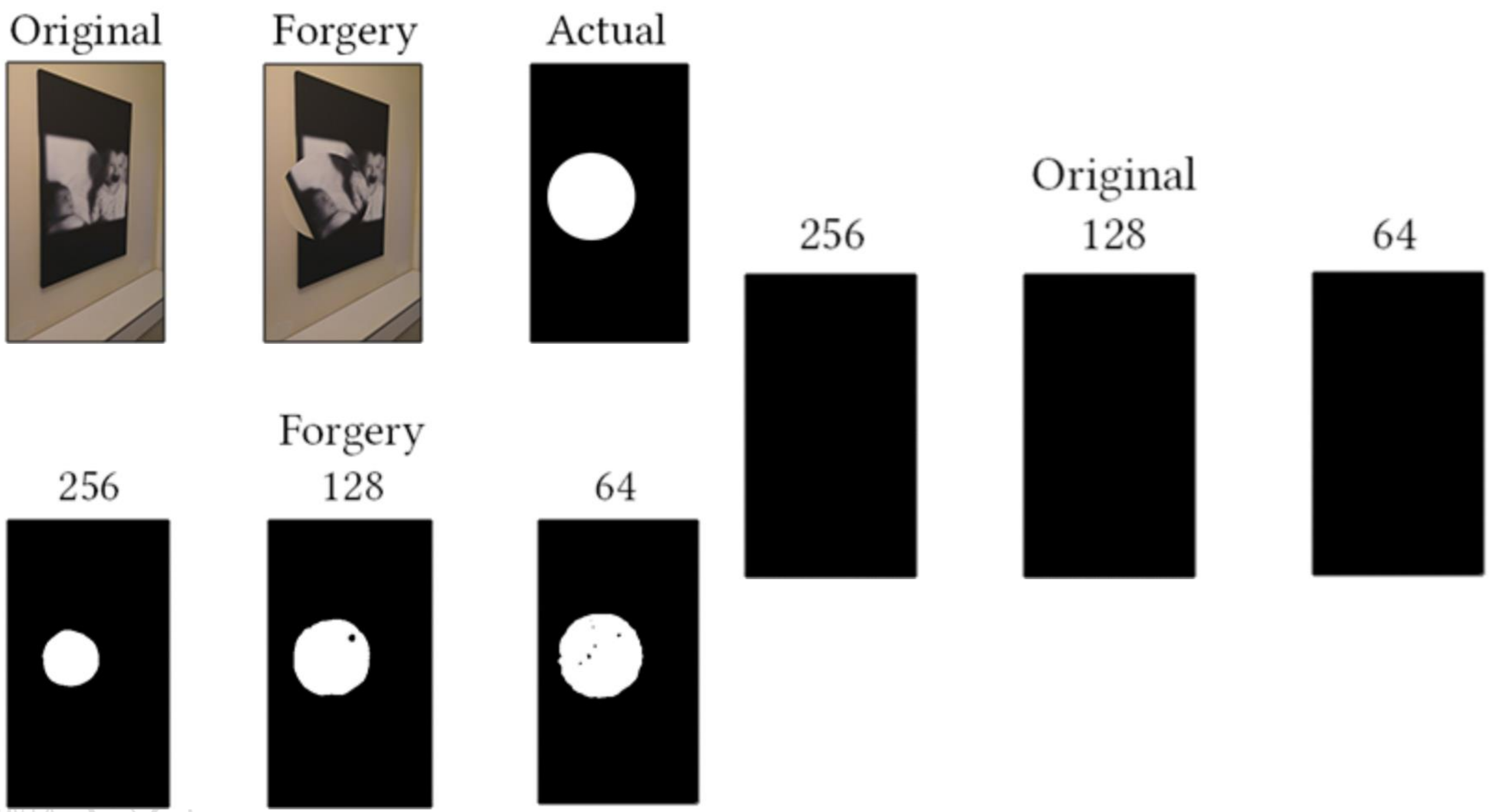


Python implementation AUC by JPEG quality.

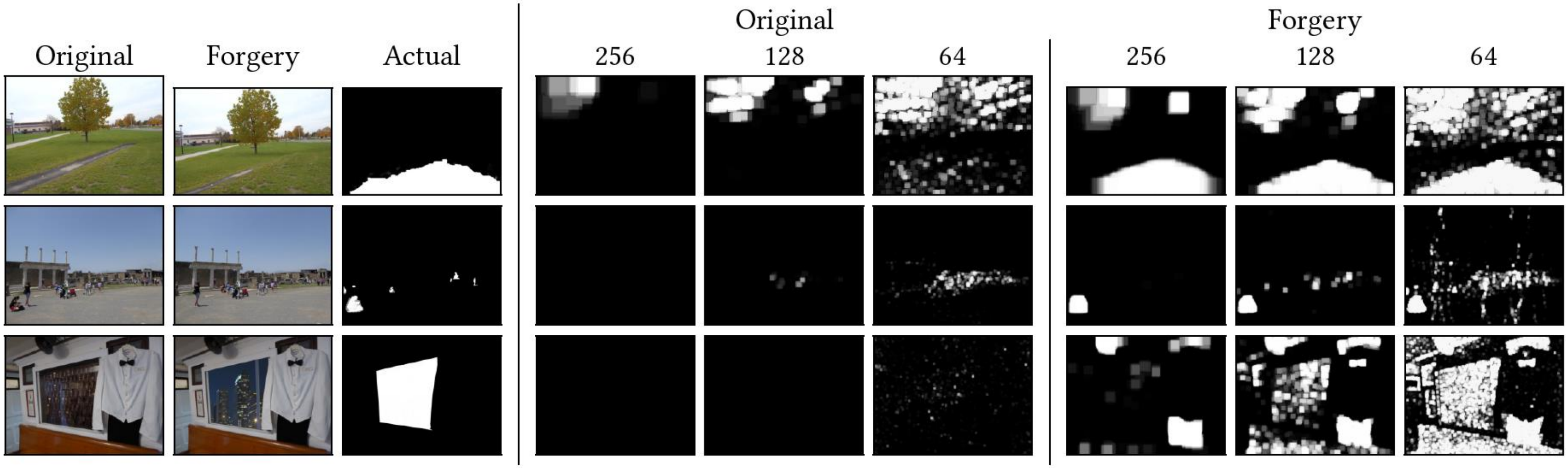
Qualitative results: paper



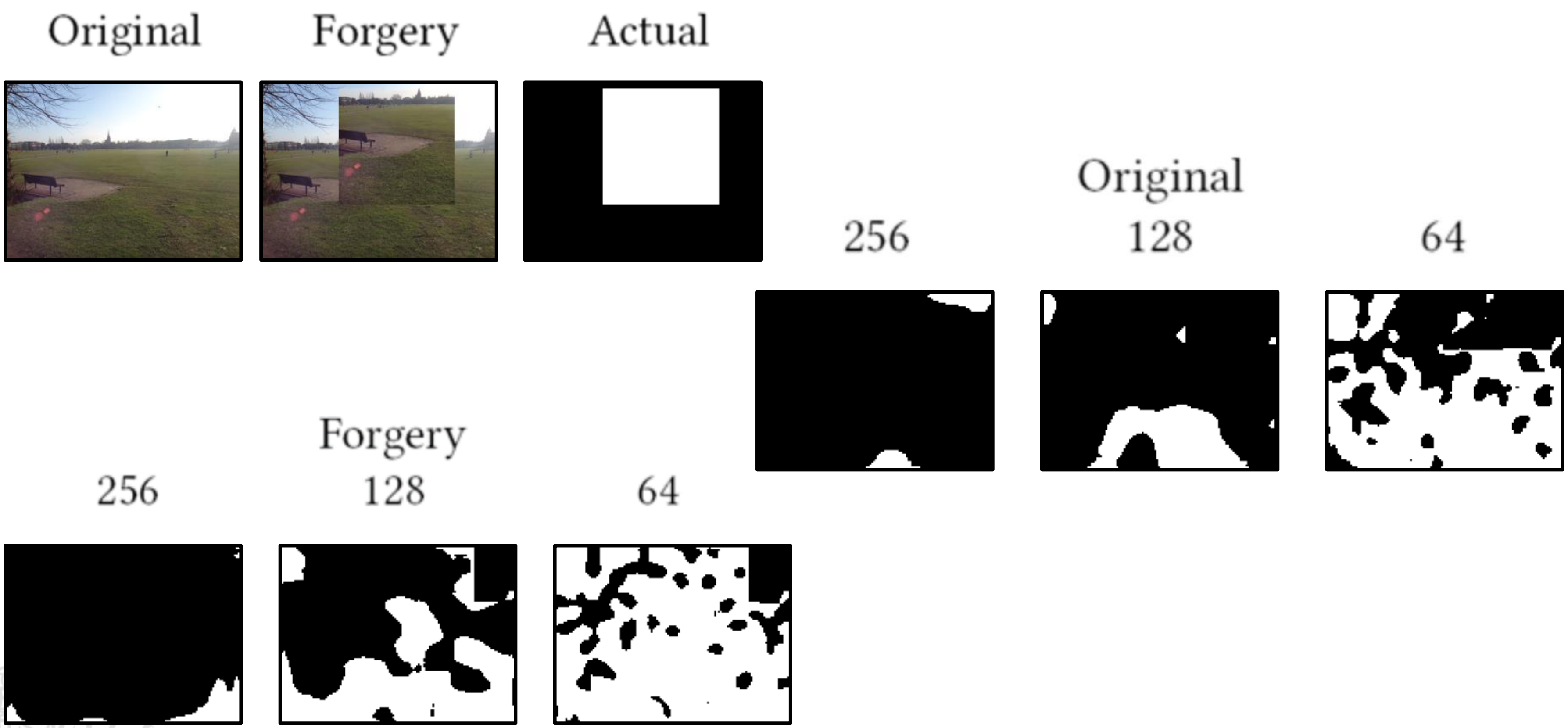
Qualitative results: Python implementation



Failure cases: paper



Failure cases: Python implementation



Conclusions

- Introduced the **JPEG compression algorithm**.
    - Focus on DCT and quantization.
- Discussed JPEG **dimple artifacts** and their use in image forensics.
- Seen a **configurable JPEG encoder** to investigate generic DCT rounding artifacts.
- Studied and implemented an **EM algorithm** for detecting popular image manipulations.
- Analyzed the algorithm's performance,
    - shown that it works quite well, and
    - has great future potential.

References

[1] S. Agarwal and H. Farid, ***Photo forensics from JPEG dimples***, 2017 IEEE Workshop on Information Forensics and Security (WIFS), 2017, pp. 1-6, doi: 10.1109/WIFS.2017.8267641.
[2] S. Agarwal and H. Farid, ***Photo forensics from rounding artifacts***, Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security, pages 103–114, 2020.
[3] A. Piva, ***Compression Standards: JPEG and MPEG***, Image Processing and Security course (Università degli Studi di Firenze, Ingegneria Informatica Magistrale), spring 2021
[4] A. Piva, ***Image Forensics: Coding-based Traces***, Image Processing and Security course (Università degli Studi di Firenze, Ingegneria Informatica Magistrale), spring 2021
[5] A. Piva, ***Image Forensics: Editing-based Traces***, Image Processing and Security course (Università degli Studi di Firenze, Ingegneria Informatica Magistrale), spring 2021

Thank you for your attention!