Contents

1 Lab 4 . . . 3
1.1 Exercise 1 . . . 3
1.2 Exercise 2 . . . 6
2 Lab 5 . . . 9
2.1 Exercise 1 . . . 9
2.2 Exercise 2 . . . 12
3 Lab 6 . . . 16
3.1 Exercise 1 . . . 16
3.2 Exercise 2 . . . 17

1. Lab 4

1.1. Exercise 1

The first exercise aims to experiment with the main lossy steps in JPEG compression - quantization.

- Load *lena512color.pgm* image and display its corresponding DCT coefficients (Fig. 1).
- Compute the compression ratio equal to the ratio of the number of all DCT coefficients from the original image to the number of kept coefficients in the compressed image (Table 2).
- Convert the compressed image into the spatial domain and display the image (Fig. 2). Compute the SSIM and MSE between the original and the reconstructed images (Table 2).
- Compress the original image using block-based approach by keeping only highest DCT coefficients in each block (Fig. 3).
- Compute the resulting compression ratio (Table 2).
- Convert the compressed image into the spatial domain and display the image (Fig. 3). Compute the SSIM and MSE (Table 2).
- Compress the original image using block-based quantization with quantization matrices of quality `50` and `90` provided in `Qtables.mat`. Convert the obtained images into the spatial domain and display them (Fig. 4). Compute compression ratios, SSIM and MSE (Table 1).

TABLE 1
Properties of the compressed images depending on the compression method.

ID	Compression Method	Coefficient Selection	**Compression Ratio**	**SSIM**	**MSE**
A	Entire image	Highest coefficients	0.0432	0.7930	90.401
B	Block-based	Highest coefficients	0.1563	0.8927	40.374
C	Block-based	Highest coefficients*	0.0469	0.7743	119.932
D	Block-based	Q_{90} matrix	0.2977	0.4849	7826.000
E	Block-based	Q_{50} matrix	0.1092	0.0796	15530.316

Observation: The best compression ratio is obtained in method **A**, furthermore, the similarity is worse by only 10% to the analogous block-based approach **B** which has the best SSIM and MSE scores. To obtain a comparable compression rate for the both **A** and **B**, the **C** method is introduced. This approach keeps fewer DCT coefficients for each block improving the compression ratio, however, this results in a similarity lower than **B**'s in terms of both SSIM and MSE.

It is important to note that although SSIM/MSE of method **D** and **E** are fairly low, the Q-matrices take advantage of psychovisual redundancies. Hence, despite low objective similairty scores the images closely resemble the original to a human eye.

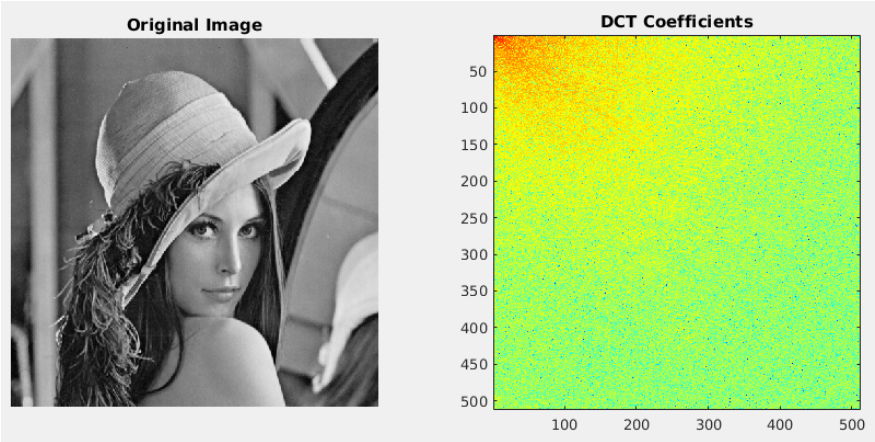

Fig 1: The original image with corresponding DCT coefficient matrix.

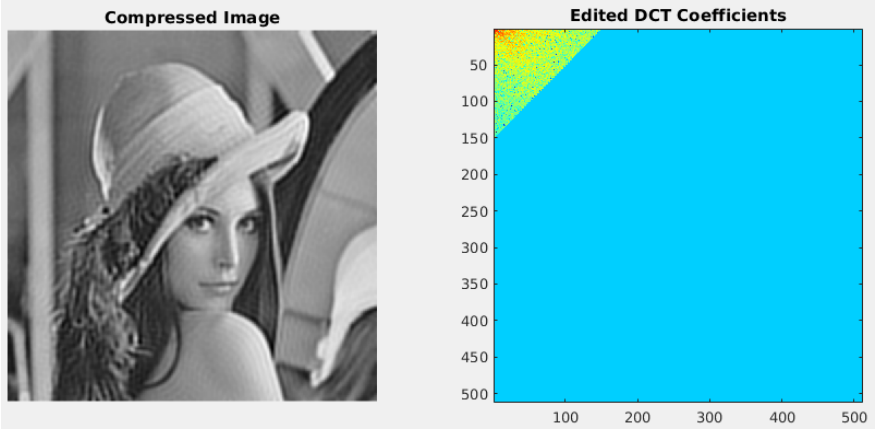

Fig 2: Reconstructed compressed image quantized by selecting highest coefficients (top-left corner of the entire image) with corresponding DCT coefficient matrix. Blue colour represents coefficients of value zero.

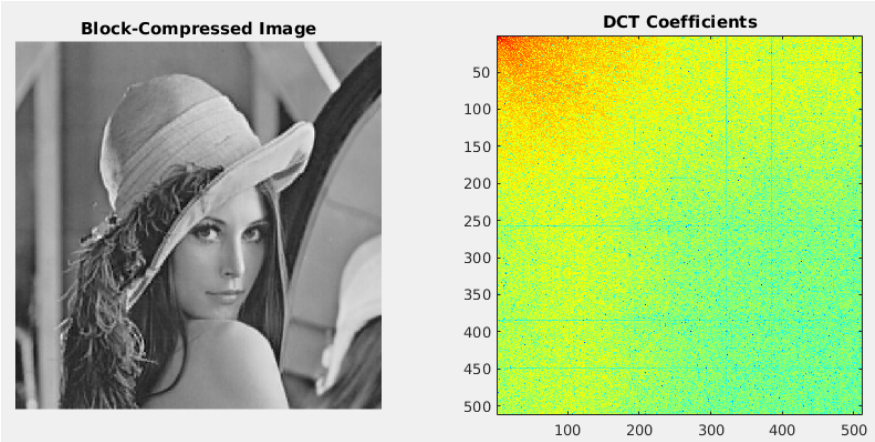

Fig 3: Reconstructed block-compressed image quantized by selecting highest coefficients (top-left corner of each block) with corresponding DCT coefficient matrix.

Fig 4: Reconstructed compressed images quantized using supplied Q_{50} and Q_{90} matrices.

1.2. Exercise 2

The goal of the second exercise of this exercise is to detect image splicing by finding JPEG Ghosts.

- Import the potentially spliced images: `'splicedBeach.jpg'`, `'splicedPlane.jpg'`, `'splicedSoldier.jpg'` and `'splicedBoat.jpg'` (Fig. 5).

- For every image generate a sequence of compressed JPEG versions with increasing quality. Calculate the difference matrix `D` between each compressed version `IQ` and the original image `I`, where an entry d_{xy} at position (x, y) is defined as:

$$d_{xy} = \frac{1}{3}\sum_{i=1}^{3}\frac{1}{b^2}\sum_{x_b=0}^{b-1}\sum_{y_b=0}^{b-1}\left[\texttt{I}(i, x+x_b, y+y_b) - \texttt{IQ}(i, x+x_b, y+y_b)\right]^2 \tag{1}$$

  where b is the block size that the values are spatially averaged from, (x_b, y_b) are relative coordinates of pixels in each such block and i corresponds to i^{th} RGB plane.

- The resulting difference matrices are then normalised using the following formula:

$$d'_{xy} = \frac{d_{xy} - min(D)}{max(D) - min(D)} \tag{2}$$

- Investigate the obtained difference images and determine whether JPEG Ghosts are present, which subsequently proves which of the original images were spliced. Results displayed in table 2.

TABLE 2
Properties of the compressed images depending on the compression method.

Image	Splicing detected	Quality of JPEG Ghost	Evidence
splicedBeach.jpg	Yes	**70** (62-76)	Fig. 6
splicedPlane.jpg	No	-	-
splicedSoldier.jpg	No	-	-
splicedBoat.jpg	Yes	**84** (78-94)	Fig. 7

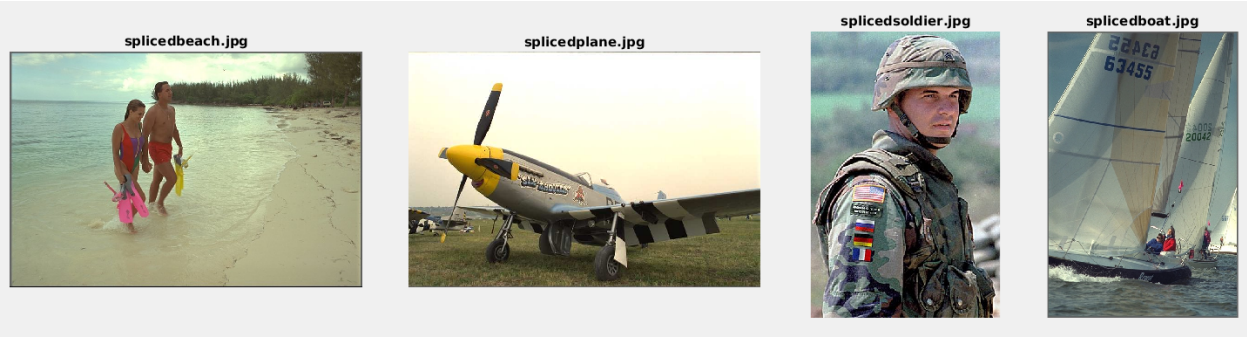

Fig 5: Original images that are to be analysed for image splicing.

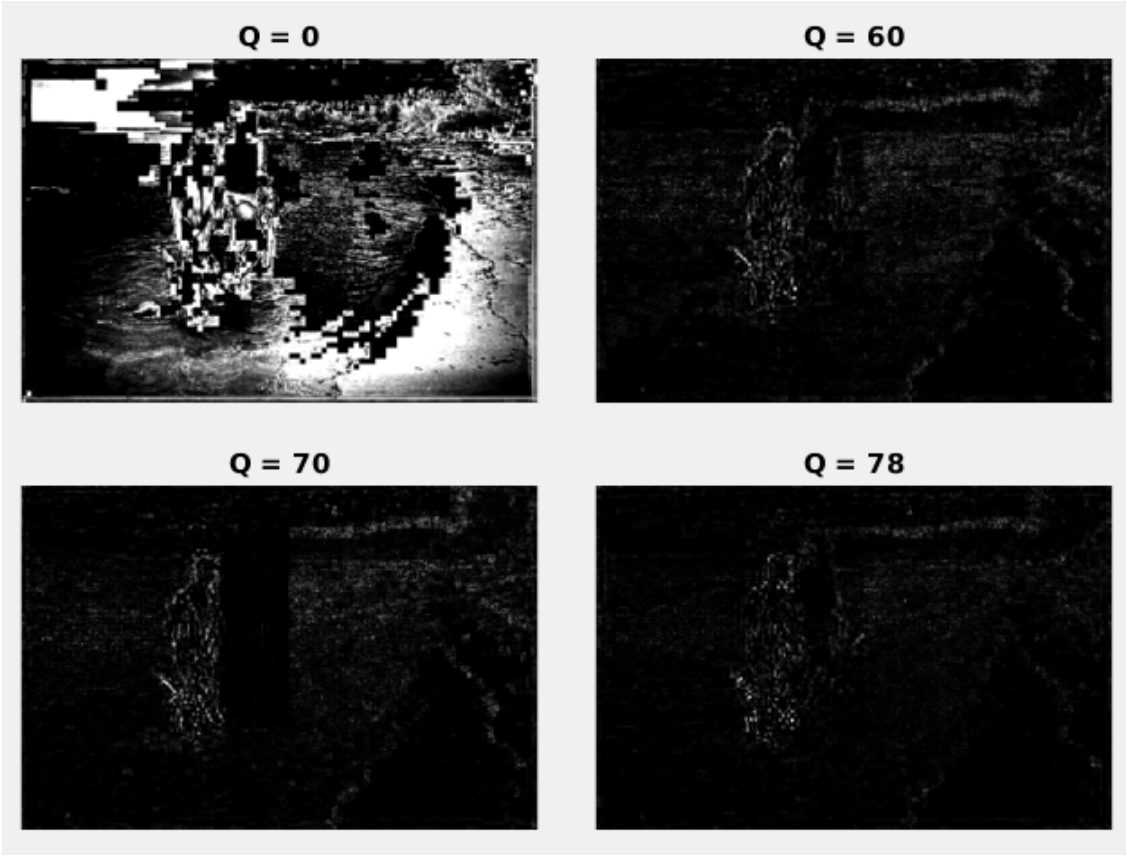

Fig 6: Evidence suggesting image splicing in file `'splicedBeach.jpg'`. A JPEG Ghost is best visible for compression quality through `60` and `78`, peaking at `70`. Detected artifacts show that the man (person on the right hand side) has been spliced.

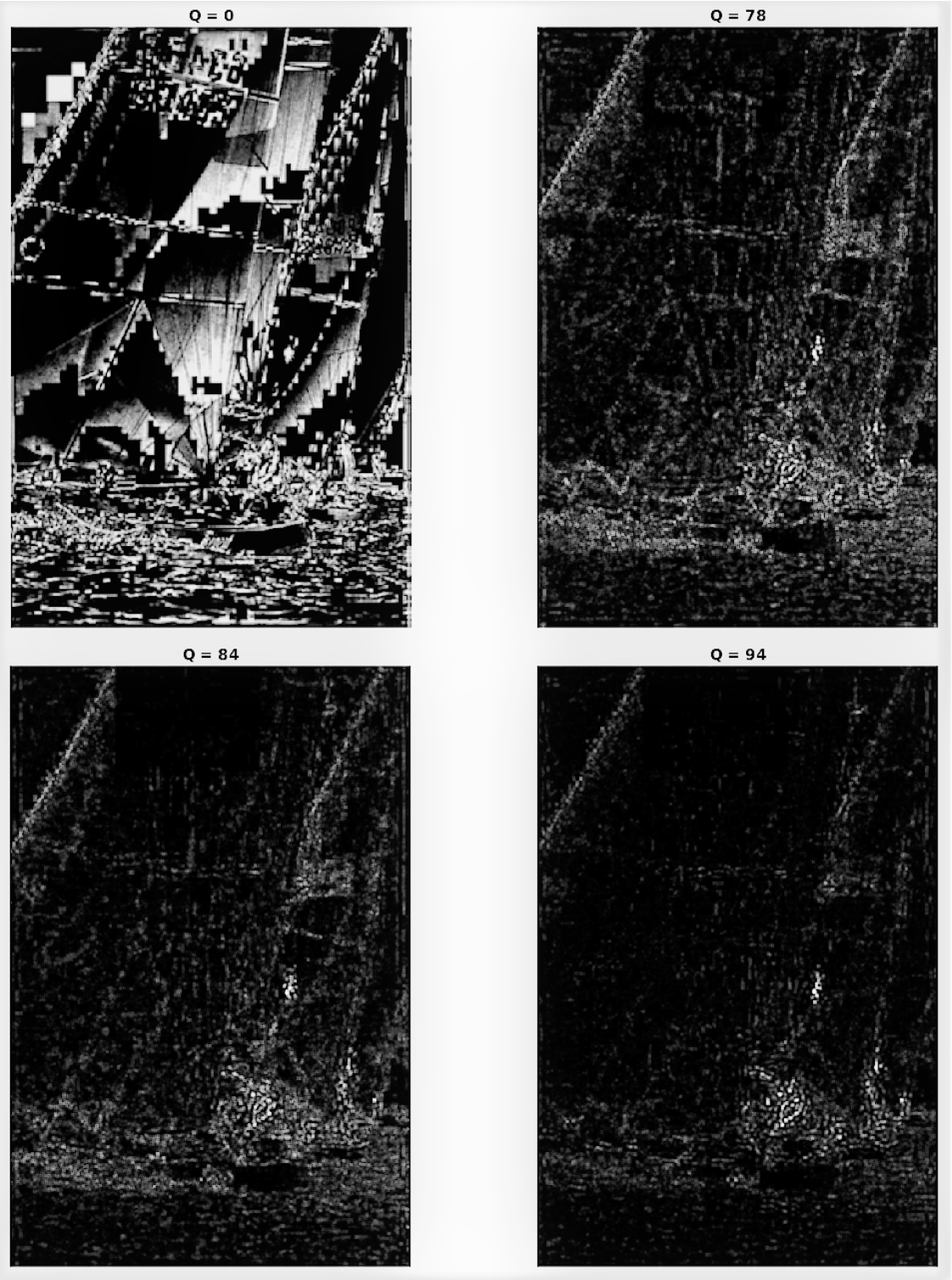

Fig 7: Evidence suggesting image splicing in file `'splicedBoat.jpg'`. A JPEG Ghost is best visible for compression quality through `78` and `84`, peaking at `94`. Detected artifacts show that the sail number (top left corner) of the boat in the foreground has been spliced.

2. Lab 5

2.1. Exercise 1

The aim of the first exercise of lab 5 was to implement the Circular Shift and Match algorithm as well as employ it to detect copy-move forgery.

- Implement the circular shift transformation (Fig. 8). Instead of calculating the new values using the formula supplied in the lectures (see CS355: Lecture 13), a faster approach that shifts not pixels but entire regions of images is utilised.
- Import the *jeep.png* image and convert it into grayscale (Fig. 9).
- Generate a circular shift image. Normalise the obtained image as well as the original image, i.e. represent each grayscale pixel as a `double` value between `0` and `1`.
- Create a function that constructs a thresholded difference image `D` between the original and the shifted image. Threshold used: `t=0.0025`
- Perform circular shift and match by adopting the algorithm proposed in the lectures (see CS355: Lecture 13). First erode, subsequently dilate the `D` image using a symmetric structuring element: `SE = strel('square', 5)`. Display the detected tempered region (Fig. 13).

Observation 1: The **similairty threshold** (`t`) must be chosen appropriately for a correct functioning of the Circular Shift and Match algorithm. Without a sufficiently discriminative `t`, the difference image will produce many false positives. On the other hand, too small `t` will result in false negatives (omitting the potentially copied regions). The effectiveness can be determined by detecting few positive responses for small shifts. The adopted threshold `t=0.0025` performs efficiently provided that the images as normalised.

Observation 2: Another key parameter is the **structuring element** (`SE`). Best performance was achieved when `SE` is: (1) identical for erosion and dilation, and (2) symmetric in shape. Moreover, it should have the size of the minimal copied region in the image that ought to considered. For this exercise, any `SE` of size < 5 detects too many false positives and `SE` of size > 15 results in numerous false negatives.

Fig 8: Circular shift method tested on the *Lena* image with different `k` (row) and `l` (column) shift values.

Fig 9: Original *jeep.png* image converted to grayscale.

Fig 10: On the left **true** regions that were copied from other parts of the image, on the right the original image prior to copy-move forgery. Source: CS355: Lecture 15.

Fig 11: On the left regions that were copied from other parts of the image, on the right the original image without the tampered regions. Here, the structuring element adopted is `strel('diamond', 12)`.

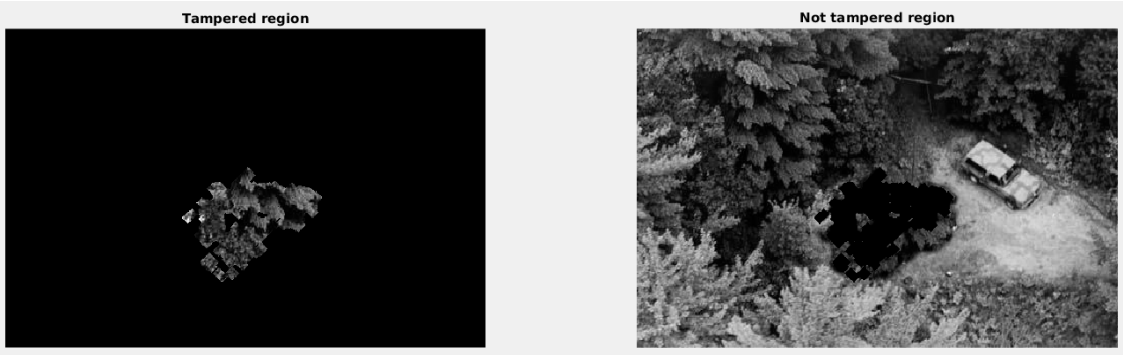

Fig 12: On the left copy-move regions, on the right the original image without the tampered regions. Here, the structuring element adopted is `strel('diamond', 5)`. This configuration produces higher granularity of the matches and possibly more false positives, however, the region of forgery is the same.

Fig 13: On the left copy-move regions, on the right the original image without the tampered regions. Here, the structuring element adopted is `strel('square', 5)`. This configuration arguably produces the best results by balancing the high granularity and number of false positives.

2.2. Exercise 2

The second exercise of this lab focused on feature extraction and matching, specifically one method described as Histogram of Oriented Gradients (HoG).

- Import the *jeep.png* image and convert it into grayscale.
- Compute the **first order gradients**: `Gx` and `Gy`, in the row and column direction (Fig. 14).
- Compute the **gradient magnitude** `Gmag` as well as the **angle gradient** `Gdir`. Wrap `Gdir` angles to fit an unsigned `0−360` degrees representation (Fig. 15).
- Generate an 18-bin Histogram of Oriented Gradients for each 8x8 block of the image (Fig. 16).
- Create a **distance matrix** of pair-wise MSE distance between histograms of each 8x8 region.
- Display 5 block pairs of the closest matches in the original image (Fig. 17).

Observation

The results obtained using this method are inconsistent with the ones from Figure 11 and expected outcome presented in the lecture. The inaccuracy of the method is potentially due to the simplicity of the feature as well as the small 8x8 block size.

It has been observed that modifying the HoG bin size or angle representation of the direction gradient (unsigned 0 to 180 and signed -180 to 180 degrees) does not yield better results for 8x8 blocks (Fig. 18). On the other hand, the increase of the block size results in significant improvement, i.e. 3 true positives (Fig. 19). Like 8x8 block approach, the change of angle representation does not affect the selection of regions (at least the top 5), however the ordering is different.

Finally, the number of bins used to compute HoG may increase the accuracy for value 36. The results are reproducible only in a small range 30-40, hence, suggesting overfitting. Since 32x32 block approach has proven to be superior, this size was adopted for bin count testing (Fig. 20)

To conclude, empirical analysis indicates that the parameters affect the practical effectiveness of HoG feature matching. Moreover, a more complex feature or variation of HoG should be considered:

- Inclusion of **colour** (or absolute intensity for grayscale) on top of sole use gradients.
- Matching **overlapping blocks** instead of non-overlapping as copied patches could be missed.

Best config (4/5 true positives): Block size = 32, bin count = 36, angle = 0 to 360/-180 to 180.

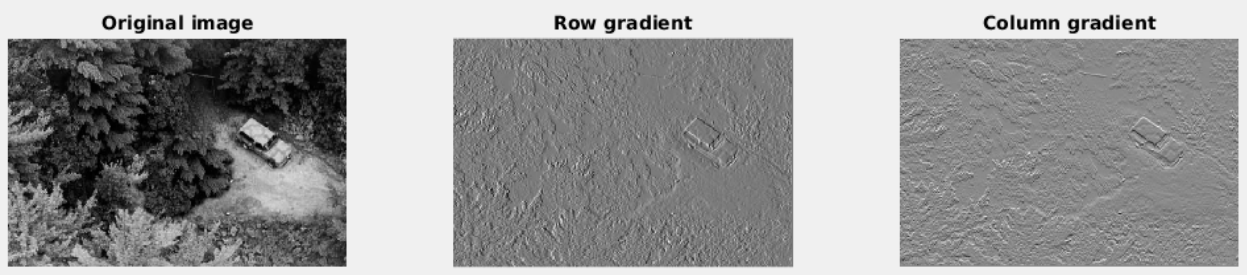

Fig 14: Left to right: the original *jeep.png* image, `Gx` row gradient of the image, and `Gy` column gradient.

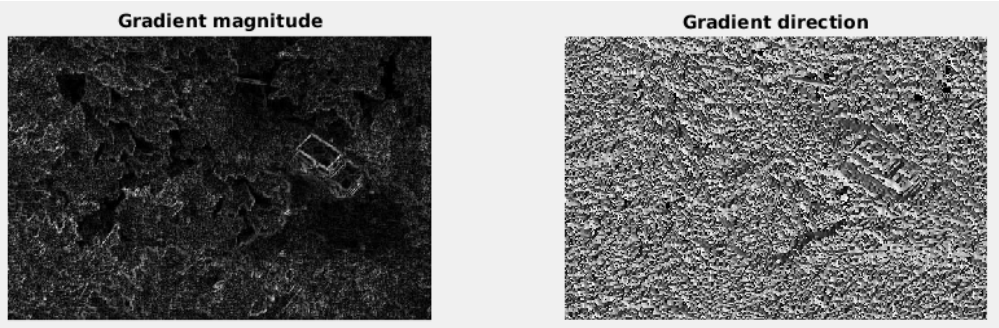

Fig 15: Left to right: `Gmag` the gradient magnitude, and `Gdir` gradient direction of the *jeep.png* image.

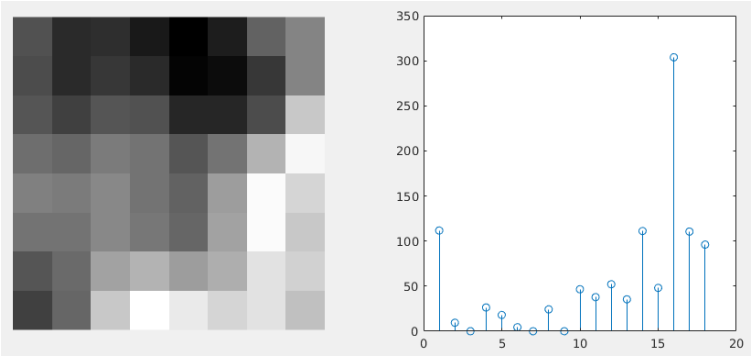

Fig 16: Randomly selected 8x8 image block with its corresponding Histogram of Oriented Gradients.

Fig 17: Five most similar pairs of 8x8 image-blocks that suggest copy-move forgery in white regions.

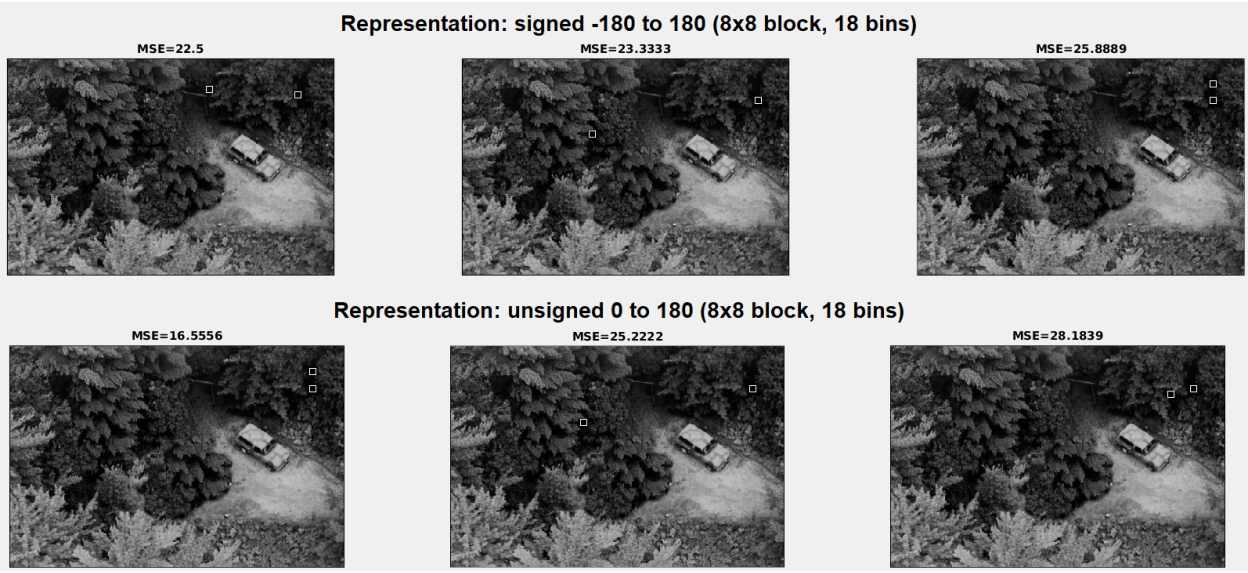

Fig 18: Three most similar pairs of blocks for each possible angle representation in `Gdir`.

Fig 19: Five most similar pairs of 32x32 image-blocks that suggest copy-move forgery in white regions.

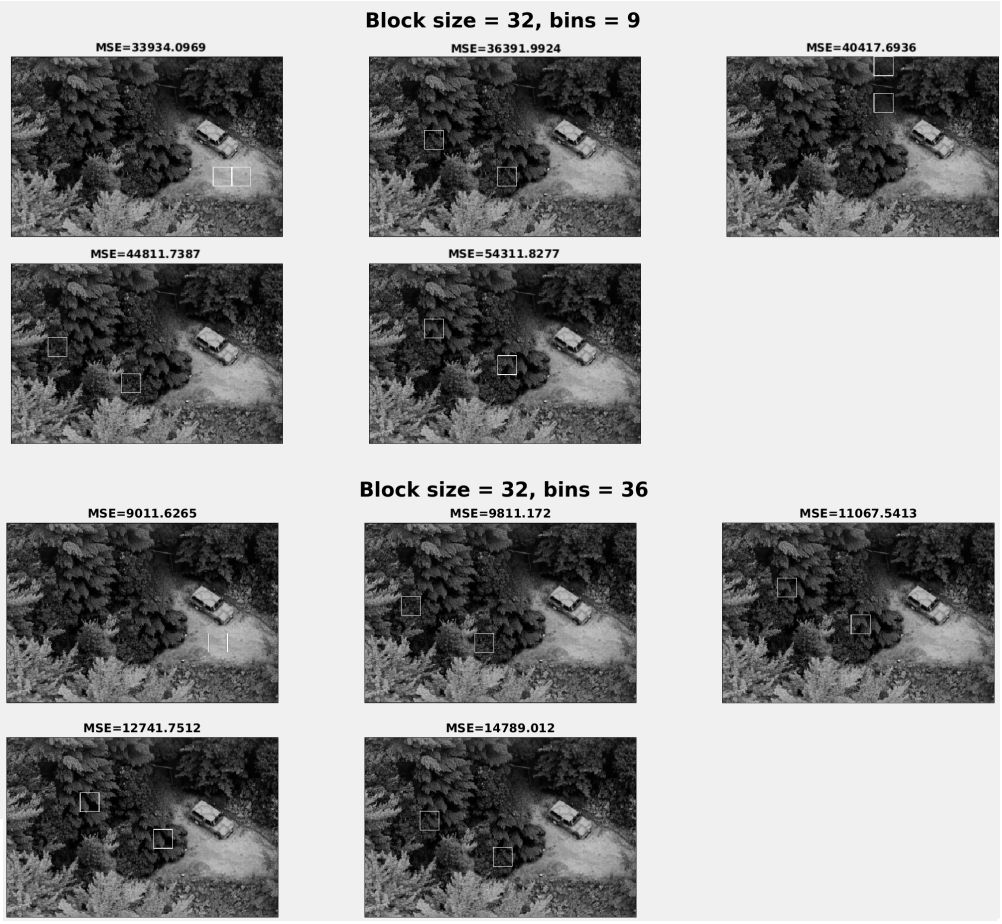

Fig 20: Five most similar pairs of 32x32 image-blocks for different number of HoG bins.

3. Lab 6

3.1. Exercise 1

The task of the first exercise in lab 6 was constructing reference SPNs from a set of reference images for 6 different camera models.

- Download the set of **reference images**. For every subfolder listed using `dir`, which represents the set of images taken by a given camera model, execute the following procedure:
  1. Enter the subfolder using the `cd()` MATLAB command.
  2. Compute SPN for each image in the subfolder by converting it into **grayscale** and **denoising** with a pixel-wise adaptive Wiener method.
  3. Crop each SPN to contain only the 800x800 region in the **centre**.
  4. **Average** the obtained SPNs to construct a final **reference SPN**.

- Save the reference SPN of each camera under a `.mat` file (lossless and allows to store in double precision) with the same name as the camera class (Fig. 21).

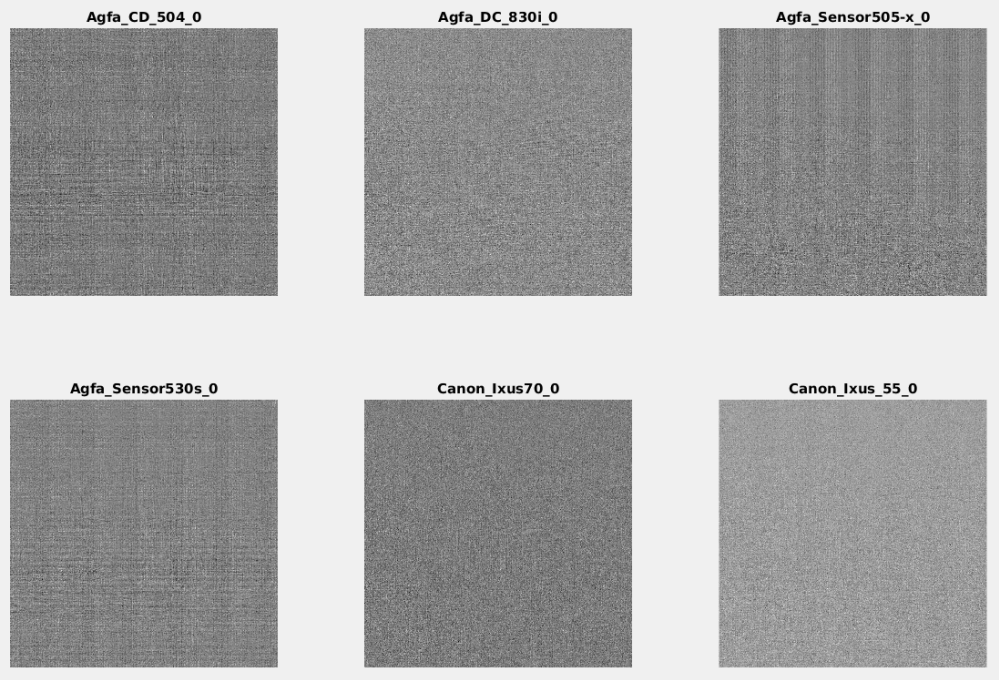

Fig 21: Reference SPNs for all six camera models considered in this exercise.

3.2. Exercise 2

The second exercise of this lab builds upon the previously computed reference SPN by employing them for source identification of a set of test images.

- Retrieve the previously calculated **reference SPNs** using the `load()` MATLAB command.
- Download the set of **training images**. For its every subfolder listed using `dir` execute the following procedure:
  1. Enter the subfolder using the `cd()` command.
  2. Compute the training SPN for each training image in the subfolder (see subsection 3.1).
  3. Calculate the **cosine similarity** between each training SPN and the reference SPN for the given camera.
  4. Select the smallest similarity as the **similarity threshold** for that camera (lower bound on an SPN match).
- Download the set of **test images**. For image listed using `dir` execute the following procedure:
  1. Compute the SPN for each test image in the folder.
  2. Calculate the **cosine similarity vector** between the test SPN and each reference SPNs.
  3. Select the camera model with the highest absolute similarity as the **match candidate**.
- Depending on the algorithm variation one can:
  (a) Use the selected candidate as the result if the minimal similarity threshold is reached, otherwise assign the image `'class 0'`.
  (b) Select the candidate regardless of the imposed similarity threshold.
  (c) *Additional*: Select the next model with the highest similarity that exceeds its threshold. If none, assign `'class 0'`.
- Display classification results (Fig. 22) and calculate the accuracy of each approach (Table 3).

Observation: The results are consistent with the intuition, i.e. if considering classifying an image as `'class 0'` is equivalent to a wrong classification, then the non-thresholded variation of the algorithm outperforms the thresholded version.

However, the contrary can be assumed in some situations - e.g. court evidence, where high certainty of the classification is required. In such a scenario, the zero classification can be counted as a correct assignment, which increases the accuracy of approach a) and b). Due to the fact that the b) method does not use class zero, its performance is not affected.

	true device	min sim	image	with threshold		without threshold		iterative threshold	
				device	similarity	device	similarity	device	similarity
1	1	0.0260	081126.JPG	1	0.0610	1	0.0610	1	0.0610
2			625619.JPG	0	0.0000	3	0.0027	0	0.0000
3			775713.JPG	1	0.0699	1	0.0699	1	0.0699
4			780228.JPG	1	0.1455	1	0.1455	1	0.1455
5			929386.JPG	1	0.0709	1	0.0709	1	0.0709
6	2	0.0185	306350.JPG	2	0.0305	2	0.0305	2	0.0305
7			435859.JPG	0	0.0000	2	0.0109	0	0.0000
8			446784.JPG	2	0.0325	2	0.0325	2	0.0325
9			486792.JPG	0	0.0000	2	0.0179	4	0.0035
10			508509.JPG	2	0.0290	2	0.0290	2	0.0290
11	3	0.0078	378610.JPG	3	0.0196	3	0.0196	3	0.0196
12			510772.JPG	3	0.0270	3	0.0270	3	0.0270
13			644319.JPG	3	0.0173	3	0.0173	3	0.0173
14			794832.JPG	3	0.0228	3	0.0228	3	0.0228
15			817628.JPG	3	0.0359	3	0.0359	3	0.0359
16	4	0.0020	350728.JPG	3	0.0636	3	0.0636	3	0.0636
17			532826.JPG	4	0.0220	4	0.0220	4	0.0220
18			811581.JPG	4	0.0210	4	0.0210	4	0.0210
19			875943.JPG	4	0.0531	4	0.0531	4	0.0531
20			939002.JPG	4	0.0603	4	0.0603	4	0.0603
21	5	0.0210	194765.JPG	5	0.0275	5	0.0275	5	0.0275
22			225922.JPG	5	0.0272	5	0.0272	5	0.0272
23			230489.JPG	0	0.0000	5	0.0142	0	0.0000
24			470924.JPG	5	0.0345	5	0.0345	5	0.0345
25			844309.JPG	5	0.0369	5	0.0369	5	0.0369
26	6	0.0132	207743.JPG	6	0.0478	6	0.0478	6	0.0478
27			301247.JPG	6	0.0238	6	0.0238	6	0.0238
28			550157.JPG	6	0.0318	6	0.0318	6	0.0318
29			587045.JPG	6	0.0366	6	0.0366	6	0.0366
30			622476.JPG	6	0.0379	6	0.0379	6	0.0379

Fig 22: Table illustrating the results of SPN matching using three approaches (left to right): **with threshold** (assigns class zero if similarity threshold is not reached), **without threshold** (assigns class with the highest similarity) and **iterative threshold** (assigns class with the highest similarity and that exceeds its similarity threshold).

Disclaimer: Class 5 and 6 are swapped around due to the alphabetical traversal of directories in MATLAB. Here, class 5 belongs to camera "Canon IXUS 70" and class 6 to camera "Canon IXUS 55".

TABLE 3
Accuracy of the three considered approaches depending on how the zero classification is treated.

Zero Classification	Method	Accuracy
Count as mismatch	With threshold	83%
	Without threshold	93%
	Iterative threshold	83%
Count as match	With threshold	96%
	Without threshold	93%
	Iterative threshold	93%