SCC0251 - Image Processing
Prof. Moacir Ponti

Final Project Report - Group 2
Detection of digital alterations in real images

Ariella Yamada Brambila - 8937034
Rodrigo de Andrade Santos Weigert - 8937503
Universidade de São Paulo
São Carlos

30/06/2017

1 What We Did

Our objective, as it was stated in the project proposal and partial report, remained unchanged. As you might recall from the partial report, we looked into various techniques we could use to fulfill that objective, but we were unsure of what to do next.

We decided to focus on one of the techniques we found, specifically the one mentioned in section 2.2 of the partial report. Please refer to it for a brief explanation of the method, which was presented by S. Bayram et al in 2005[1]. We sought to reproduce some of the results of his study, except for a couple of important differences:

- For selection of the best features of the images, we didn't use the Sequential Floating Forward Search algorithm, with which we aren't familiar. Instead we used Principal Component Analysis (PCA) to transform the feature space and measure the relevance of each post-transformation feature, which allowed us to discard the least relevant ones.
- We used a *much* larger dataset. While the original study used only a coulpe of hundred of images, we worked with the over twelve thousand images of the CASIA v2 dataset[2]. CASIA v2 contains 7.491 authentic and 5.123 tampered images of very diverse content (from nature scenes to textures and patterns). The tampered images are all derived from the authentic ones, and the tampering process underwent to each one is described somewhat accurately by the names of the image files.

Unfortunately due to time constraints we couldn't implement every single one of the tests perfomed by Bayram, but the tests we performed did get interesting results nevertheless.

2 How We Did It

With a rather unsettling amount of Python 3 scripts (with numpy and OpenCV for all the image processing) and also some C++ code. The general idea will be presented in the following paragraphs, but the precise details of how to reproduce everything we did is on the various README files present in the GitHub repository.

First, we created a python module *bsm.py* to calculate the same Binary Similarity Measures (BSMs) of the original study. If we understood it correctly, there were exactly 102 measures per image:

- 9 measures for each of the bit planes 3, 4, 5, 6, 7 and 8 of the red channel of the image, which totalizes 54. Note: Bayram numbers the bit planes from 1 to 8. We assumed that bit plane 1 is the one related to the least significant bit of the pixel value.
- 8 measures for each of the bit plane pairs 3-4, 4-5, 5-6, 6-7, 7-8 of the red channel, totalizing 40 measures.
- 8 measures for the bit plane pair 5 (red channel)-5(blue channel).

We then created *extractFeatures.py* - the script that uses *bsm.py* to extract the features of all images in a given directory, and *processData.py* - which is used to apply the PCA, select the N most relevant features (we used different N for different tests, but $N = 32$ for all the final tests presented in the next section) and divide the images into the training and test sets for the classifier. There are a couple of other scripts involved in the process. Here's a brief description about them:

- *analyzeData.py*: used to count the instances of tampered and authentic images contained in the random samples we generate. This information is useful when evaulating the quality of the classifiers we obtain.
- *randomSample.py*: to obtain random samples from the CASIA v2 dataset.
- *removeNames.py*: to replace the names of the images *extractFeatures.py* outputs in the first column of the features file with their respective labels/classes (0 for authentic, 1 for tampered)
- *separateImages.py*: to select images from the CASIA v2 dataset which underwent a specific form of tampering based on the name of the files.

Finally we have the classifier, which is our own C++ implementation of a multilayer perceptron (MLP) input layer of size N, one hidden layer and output layer of size 1 (output 0 represents an authentic image, 1 represents a tampered one). For more details on the classifier, please read the file *about_the_classifier.txt*.

3 Results

3.1 All CASIA v2

Our first test was using the entire unmodified CASIA v2 dataset through our classifier. It's no surprise the results obtained weren't great. We got 63.5% accuracy, while the majority class (authentic images) represented 60% of all images. This result is expected, as this classifier isn't powerful enough to perform well on this highly nontrivial classification task.

Table 1: Confusion Matrix for All CASIA v2 test

		Predicted Class	
		Authentic	Tampered
Real Class	Authentic	1662 (68.23% of all negatives)	774 (31.77% of all negatives)
	Tampered	604 (44.81% of all positives)	744 (55.19% of all positives)

3.2 CASIA D 3000

In this test we selected all images in the CASIA v2 dataset that had content from a different image pasted on them, which amount to about 1800. To balance the amount of authentic images in the set, we randomly selected only 1500 of them. After processing, we trained the classifier with 70% of images from this set and obtained 65.4% of accuracy in the test with the remaining 30% , which is only marginally better than the performance on the All CASIA v2 test.

3.3 Gaussian 1000

Our best result was obtained by using the *randomSample.py* to select 500 images from the authentic set of CASIA v2 and then blurring them significantly with a Gaussian filter, creating a dataset of 1000 images. The obtained classifier got a 96.6% of accuracy, using half the images for testing and half for training.

Table 2: Confusion Matrix for CASIA D 3000 test

		Predicted Class	
		Authentic	Tampered
Real Class	Authentic	251 (63.07% of all negatives)	147 (36.93% of all negatives)
	Tampered	201 (33.11% of all positives)	406 (66.89% of all positives)

Table 3: Confusion Matrix for Gaussian 1000 test

		Predicted Class	
		Authentic	Tampered
Real Class	Authentic	236 (96.33% of all negative)	8 (3.14% of all positives)
	Tampered	9 (3.67% of all negatives)	247 (96.86% of all positives)

1

3.4 Increase Scale 1000

For this test we did almost exactly the same as in the previous one, except we used scaling by a factor of 2.0 in both directions instead of Gaussian blur. We obtained 80.2% of accuracy in the test set.

Table 4: Confusion Matrix for Increase Scale test

		Predicted Class	
		Authentic	Tampered
Real Class	Authentic	193 (86.94% of all negatives)	29 (13.06% of all negatives)
	Tampered	70 (25.18% of all positives)	208 (74.82% of all positives)

3.5 Reduce Scale 1000

In this test we did exactly the same in the previous test, except with scaling factor of 0.5. The results obtained were a bit lower than the opposite scaling test. The accuracy was obtained 72%.

Table 5: Confusion Matrix of Reduce Scale 1000 test

		Predicted Class	
		Authentic	Tampered
Real Class	Authentic	163 (75.12% of all negatives)	54 (24.88% of all negatives)
	Tampered	84 (29.68% of all positives)	199 (70.32% of all positives)

4 On the Results

We considered the results very good. They do offer further evidence that the Binary Similarity Measures may be able to provide techniques that can reach our original goal with respectable accuracy. In our opinion they are definitely worth of more exploration and testing with different, possibly fancier techniques, like building different classifiers for different kinds of tampering and then creating an ensemble, which was (again, as far as we understood) only superficially explored by Bayram[1]. Another path of possible exploration would be trying to locate the tampering in some way, maybe with something as simple as just dividing the test image in smaller blocks and running each block through the classifier.

Although we regret we couldn't try and do more tests and ideas, we are satisfied with the results and think what we did isn't very far from the best we could do given our proposal and our time and knowledge constraints.

References

[1] S. Bayram et al. “Image manipulation detection with Binary Similarity Measures”. In: *2005 13th European Signal Processing Conference*. Sept. 2005, pp. 1–4.

[2] *CASIA Tampered Image Detection Evaluation Database*. `http://forensics.idealtest.org/casiav2/`. Accessed: 2017-28-06.

Credits for the use of the CASIA Image Tempering Detection Evaluation Database (CAISA TIDE) V2.0 are given to the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science, Corel Image Database and the photographers. `http://forensics.idealtest.org`