**Stato dell’arte – MOD4**

***Parametri estrazione IMs (GGE e IIC)***

Partendo dall’intuizione nell’articolo di Victor e Tiago (*Exploring Statistical Differences Between Illuminant Estimation Methods for Exposing Digital Forgeries*) che, data un’immagine vera ed una contenente una contraffazione, la differenza tra le due Illuminant Maps estratte (GGE e IIC) è maggiore in quella contraffatta, ho innanzitutto provato a vedere se fosse giustificata non solo a livello visivo (come detto nell’articolo), ma anche a numericamente.

Il dataset di riferimento è il DSO-1, passatomi da Tiago. Tuttavia il dataset in mio possesso non è completo a causa di una parte di archivio corrotto: non ho dunque alcune immagini, ma soprattutto non ho il groundtruth di riferimento per la valutazione delle performance (ho solamente l’associazione dell’intera immagina alle classi NORMAL e SPLICING, mentre nell’articolo si parla di ROI di riferimento).

Per ciascuna immagine del dataset ho estratto le due mappe GGE e IIC (il procedimento di estrazione delle due mappe è lo stesso usato da Tiago nel suo articolo *Illuminant-Based Transformed Spaces for Image Forensics (*e quindi non quello effettivamente usato per l’articolo di Victor e dunque specifico per le facce) dalle immagini intere e calcolata la distanza Euclidea (in RGB) pixel-wise tra le due immagini. A questo punto sommando tutti gli elementi della matrice delle distanze ottengo un valore numerico che associo alla distanza tra le intere immagini. In accordo con la precedente intuizione, la distanza dovrebbe essere maggiore per le immagini appartenenti alla classe SPLICING rispetto a quelle NORMAL. Questo però non si verifica utilizzando le mappe estratte con i parametri presenti nei codici sorgenti di Tiago: chiedendo spiegazioni in merito direttamente a lui e a Victor mi è stato risposto che questi parametri probabilmente sono stati trovati in riferimento alle facce, non a generiche immagini.

Per questo motivo sto provando attualmente a cercare un tuning dei parametri di estrazione delle due mappe (ripartendo da quelli di default) che massimizzi la separazione tra le due classi utilizzando come discriminante il valore della differenza tra GGE e IIC calcolata come in precedenza.

***Analisi del feature vector***

Prima di procedere con la fase di addestramento (mi sono concentrato sul dataset fornito da Tiago per avere un riscontro a livello di performances), ho provato ad analizzare i feature vector generatati per verificare se una delle dimensioni risultasse maggiormente discriminante rispetto alle altre.

L’ipotesi era quella di esplorare le dimensioni, analizzandole con una curva ROC, per poi, in caso assegnare un peso diverso alle diverse dimensioni in base al suo potere discriminativo.

I risultati però evidenziano che, allo stato attuale, nessuna dimensione ha questa caratteristica.

Nello zip in allegato sono presenti tutte le curve ROC generate per gli autovalori.

***Algoritmo di detection***

L’algoritmo allo stato attuale funziona nel seguente modo (fase di detection):

1. Data un’immagine ne estrae le due Illuminant Maps (GGE e IIC).
2. Le due mappe vengono convertite in scala di grigio, in quanto, considerandole come un nuovo spazio colore. Questo a seguito di uno scambio di mail con Tiago:

*“When dealing with eigenvalues descriptor, I just considerate the maps intensity. So, I simplify the problem converting the illuminant image to gray scales.*

*Along my researches using illuminant maps, I learned that, most of times, the illuminant color (practically and empirically speaking) can be considered but it is not the main part for splicing detection. For a better understanding, you can imagine that illuminant map is another color space (as YCbCr, for example). So, maybe you can try split the illuminant images channels and test the technique with each one of them.”*

1. Viene creata una matrice delle stesse dimensioni dell’immagine originale. Il valore del pixel alla posizione (x, y) indicherà lo score del pixel selezionato: tanto più questo sarà alto, tanto più sarà probabilmente stato manomesso. Questa tecnica, proposta da Victor, si basa sul fatto che i pixel con la ricorsione di cui sotto vengono esaminati più volte: più volte un pixel sarà stato incluso in una porzione di immagine ritenuta falsificata, tanto più è probabile che faccia parte di una regione effettivamente modificata.
2. Inizia la ricorsione. L’immagine viene divisa in 4 (2 tagli uniformi lungo i due assi) e per ciascun blocco:
   1. Calcola gli autovalori per ciascuna delle due porzioni di mappetramite SVD e ne considera solo i primi 9 principali.
   2. A partire dagli autovalori estratti viene generato il feature vector formato dalla concatenazione delle differenze statistiche tra coppie di autovalori (di default viene considerata la distanza logaritmica)
   3. Il blocco viene classificato come vero o falso tramite classificazione binaria (o regressione se specificato)
   4. Vengono incrementati i valori dei pixel corrispondenti al blocco in esame del risultato della classificazione.
3. Al termine della ricorsione (si imposta un massimo numero di iterazioni) in base ad un valore di soglia viene stabilito quale regione dell’immagine è stata manomessa in base ai valori della matrice. La regione così definita potrà assumere forme variegate e non solo rettangolari.

***Codice sorgente***

Il codice sorgente è attualmente caricato su GitHub, in modo tale che possiate vedere lo stato di avanzamento dei lavori. Il codice si trova alla pagina:

https://github.com/lorecioni/ImageSplicingDetection

**TODO List**

* Valutare accuracy sul Columbia Dataset usando i parametri di Riess
* Capire come Riess estrae la mappa IIC (l’algoritmo)
* Mail a Tiago: capire come mai la mappa estratta è diversa da quella che riesco ad estrarre io. Qual è la segmentazione utilizzata in origine.

**Image splicing detection – Codice sorgente**

Linguaggio utilizzato Python, versione 3.4.

Librerie utilizzate: OpenCV 3.0

Tutto il codice sorgente è completamente aperto.

https://github.com/lorecioni/ImageSplicingDetection

*Generazione delle Illuminant Maps*

Linguaggio C++

<https://www5.cs.fau.de/en/research/areas/computer-vision/image-forensics/scene-illumination-as-an-indicator-of-image-manipulation/>

*Estrazione dei descrittori*

Linguaggio C++

ACC, BIC, CCV, EOAC, LAS, LCH, SASI, SPYTEC, UNSER

**Funzione principale: main.py**

ImageSplicingDetection v.0.1

Creation date: 2016-09-12, last update: 2016-11-14

usage: main.py [-h] [--train] [--detect] [--cross-validate]

[--extract-single-features] [--no-extract-features]

[--no-extract-maps] [--img IMG] [--heat-map] [--verbose]

optional arguments:

-h, --help show this help message and exit

--train train the model for further splicing detection

--detect detect splice over an image

--cross-validate cross-validate the dataset

--extract-single-features extract feature vector for a specific image

--no-extract-features no extract all training images features

--no-extract-maps no extract all training images features

--img IMG the path of the suspicious image

--heat-map display the heat map between GGE and IIC maps

--verbose display all messages