**Image splicing detection**

L’algoritmo sviluppato di detection di falsificazioni all’interno di un’immagine è composto da due moduli.

1. *Face splicing detection*
2. *Region splicing detection*

Ciascuno dei due moduli esegue in autonomia sull’immagine data in ingresso e restituisce:

* Una **label** associata all’immagine (*PRISTINE, FAKE*)
* Uno **score**, o grado di confidenza della predizione
* Una **mappa** delle parti dell’immagine ritenute false, se classificata come *FAKE*.

# Face splicing detection

L’algoritmo, presentato in [1], consente la classificazione di un’immagine contenente almeno due facce come autentica o falsificata. In particolare è possibile andare ad individuare, nel caso di un’immagine falsa, quale delle facce presenti nell’immagine è stata manomessa.

## Algoritmo

Viene descritta a seguito la procedura implementata per il **training**.

1. **Illuminant maps extraction**

Per ciascuna immagine del dataset, estrae la GGE e la IIC map dell’intera immagine. Le due mappe vengono considerate come un nuovo spazio colore dell’immagine originale. L’estrazione delle mappe viene effettuata con la tecnica descritta da Riess in [2].

1. **Face detection and paired face feature extraction**

Eseguita l’estrazione delle due mappe, viene letto dal groudtruth la posizione delle facce.

Se l’immagine contiene un numero maggiore di due facce si continua con il processo, altrimenti l’immagine viene scartata.

Per ciascuna faccia presente, si ricavano le due mappe (GGE e IIC) relative a quest’ultima (selezionando la regione nelle stesse corrispondente) e si esegue un algoritmo di estrazione di un descrittore di colore. I descrittori presi in considerazione sono il ACC, BIC, CCV e LCH.

A questo punto per ciascuna coppia di facce presenti ne viene creato un descrittore concatenandone i due descrittori: una coppia sarà considerata come falsificata se almeno una delle due facce presenti risulta essere falsa, altrimenti sarà considerata come autentica.

1. **KNN models training**

Per la fase di training viene innanzitutto selezionato un valore di K per l’addestramento dei modelli KNN.

Per ciascun diverso descrittore viene poi creato un modello (i.e. un modello che utilizza descrittori AAC, uno per BIC, etc.).

Viene descritta a seguito la procedura implementata per la **detection**.

1. **Illuminant maps extraction**
2. **Face detection and paired face feature extraction**

Come nel caso precedente, ma la detection delle facce viene effettuata tramite l’algoritmo di Viola e Jones, con la sua implementazione per OpenCV.

1. **KNN classification**

Ciascuna coppia di facce viene ora classificata tramite i modelli KNN addestrati. Per ciascun classificatore otteniamo una prediction, il risultato è ottenuto tramite votazione a maggioranza tra tutti i classificatori: una coppia di facce sarà considerata come falsa se almeno la metà più 1 dei classificatori utilizzati la classifica come non autentica.

1. **Single face choose**

Data una coppia di facce considerata falsa, si procede alla scelta della faccia falsa tra le due. Questo viene fatto tramite un classificatore a singola faccia addestrato sullo stesso dataset.



Nell’immagine precedente è mostrato l’output della procedura illustrata su un’immagine falsificata (la donna sulla destra). Si verifica qua un caso di falso positivo, in quanto anche la figura di sinistra viene classificata come falsa, anche se con score diversi: quella di destra ha uno score del 75%, quella di sinistra del 65%.

# Region splicing detection

Il secondo modulo consiste in un detector di regioni dell’immagine falsificate [4]. Questo particolare approccio è di tipo *blind,* non ha bisogno di *human intervention* ed è adatto a immagini dal generico contenuto (non è dunque vincolato alla presenza di facce nell’immagine come nel caso precedente).

Non è presente una fase di training, è però necessario indicare a priori alcune delle soglie (punto critico dell’algoritmo è proprio la selezione di tali soglie).

## Algoritmo

1. **Local illuminant estimation**

L’immagine viene suddivisa in bande orizzontali e verticali. Viene fissata a priori la dimensione (orizzontale e verticale) in base alla dimensione stimata degli oggetti di interesse. La stima dell’illuminant map viene effettuata usando 5 diversi algoritmi: Gray-World, Max-RGB, Shades of Gray, first order Gray-Edge e second order Gray-Edge. In pratica questa fase è divisa in due step consecutivi:

* 1. ***Suddivisione dell’immagine in bande.***
  2. ***Per ciascuna banda, stima dell’IM utilizzando ciascuno dei 5 diversi algoritmi.***

1. **Region splicing detection**
   1. ***Reference illuminant estimation***

Per ciascun algoritmo utilizzato in precedenza, vengono adesso stimati i due colori di riferimento (uno per le bande verticali ed uno per quelle orizzontali. Il colore stimato è dato dalla mediana dei colori di una singola direzione per uno specifico algoritmo.

* 1. ***Distance eveluation***

Vengono ora calcolate le distanze euclidee tra il colore di riferimento (per ciascun algoritmo e per ciascuna direzione) ed il colore stimato della singola banda. Se la distanza è maggiore di una soglia prefissata, la banda viene marcata come sospetta.

* 1. ***Detection map***

Per identificare le regioni sospette viene costruita una detection map. Ogni volta che una regione è classificata come falsa (vedi punto precedente), si incrementano di un’unità i pixel di una mappa corrispondenti a quelli della regione in esame.

* 1. ***Output***

Data la detection map viene ora sogliata per ottenere la maschera del tampering predetto.



Nell’esempio mostrato in figura viene mostrata a sinistra l’immagine falsificata, a destra l’output della detection map. Nell’immagine di sinistra, la persona al margine destro è stata aggiunta a posteriori.

La **detection map** è così interpretabile: a colori freddi corrisponde un basso score di detection, a colori caldi un alto score di detection. E’ possibile notare che i valori più alti si hanno proprio nelle bande verticali corrispondenti alla persona falsificata sulla destra.

Sogliando la detection map (prendendo solamente detection con valori superiori all’80% di confidenza) otteniamo il seguente risultato:



Nel caso precedente la confidenza risultante era del *95%*.

# Dataset

I dataset utilizzati per i test del procedimento sono il DSO-1 e il DSI-1, entrambi reperibili in rete e creati da Tiago.

<https://recodbr.wordpress.com/code-n-data/#dso1_dsi1>

DSO-1*It is composed of 200 indoor and outdoor images with an image resolution of 2,048 x 1,536 pixels. Out of this set of images, 100 are original, i.e., have no adjustments whatsoever, and 100 are forged. The forgeries were created by adding one or more individuals in a source image that already contained one or more persons.*

DSI-1*It is composed of 50 images (25 original and 25 doctored) downloaded from different websites in the Internet with different resolutions. Original images were downloaded from Flickr and doctored images were collected from different websites such as Worth 1000, Benetton Group 2011, Planet Hiltron, etc.*

Il groundtruth è composto da un file di testo per ciascuna immagine così popolato:

*1 FAKE 1112 1502 496 952*

*2 NORMAL 680 1100 388 890*

*…*

In ordine di apparizione:

* *Il numero della faccia nell’immagine (progressivo)*
* *La classificazione (FAKE o NORMAL)*
* *Top left x*
* *Top left y*
* *Bottom right x*
* *Bottom right y*

# Esperimenti

1. **Classificazione con crossvalidazione coppie di facce (ACC) (10/01/2017)**

Dataset: DSO-1

Descrittori: ACC

Mappe colore: GGE (1 order).

Accuracy (avg over 5 runs): 70.2%

1. **Classificazione con crossvalidazione coppie di facce (ACC e BIC) (11/01/2017)**

Training e valutazione delle performance con crossvalidazione (10-fold) del DSO-1 utilizzando un solo classificatore addestrato con descrittori ACC e BIC.

Dataset: DSO-1

Descrittori: ACC, BIC

Mappe colore: GGE (1 order)

Accuracy (avg over 5 runs): 72.2%

1. **Classificazione con crossvalidazione coppie di facce (ACC, BIC, CCV) (11/01/2017)**

Training e valutazione delle performance con crossvalidazione (10-fold) del DSO-1 utilizzando un solo classificatore addestrato con descrittori ACC, BIC e CCV.

Dataset: DSO-1

Descrittori: ACC, BIC

Mappe colore: GGE (1 order)

Accuracy (avg over 5 runs): 73.2%

1. **Classificazione con crossvalidazione coppie di facce (ACC, BIC, CCV e LCH) (16/01/2017)**

Training e valutazione delle performance con crossvalidazione (10-fold) del DSO-1 utilizzando un solo classificatore addestrato con descrittori ACC, BIC, CCV e LCH.

Dataset: DSO-1

Descrittori: ACC, BIC, CCV, LCH

Mappe colore: GGE (1 order)

Accuracy (avg over 5 runs): 75.6%

1. **Classificazione con crossvalidazione coppie di facce (ACC, BIC, CCV e LCH) (20/01/2017)**

Training e valutazione delle performance con crossvalidazione (10-fold) del DSO-1 utilizzando un solo classificatore addestrato con descrittori ACC, BIC, CCV e LCH.

GGE e IIC map.

Dataset: DSO-1

Descrittori: ACC, BIC, CCV, LCH

Mappe colore: GGE (1 order), IIC

Accuracy (avg over 5 runs): 77.6%

1. **Classificazione con crossvalidazione coppie di facce (ACC, BIC, CCV e LCH) (20/01/2017)**

Training e valutazione delle performance con crossvalidazione (10-fold) del DSO-1 utilizzando un solo classificatore addestrato con descrittori ACC, BIC, CCV e LCH. Con descrittori inveriti.

GGE e IIC map.

Dataset: DSO-1

Descrittori: ACC, BIC, CCV, LCH

Mappe colore: GGE (1 order), IIC

Accuracy (avg over 5 runs): 77.9%

1. **Classificazione DSI-1**

Dataset: DSI-1

Descrittori: ACC, BIC, CCV, LCH

Mappe colore: GGE (1 order)

Accuracy: 82.2%

1. **Classificazione DSI-1, training su DSO-1**

Dataset: DSI-1

Descrittori: ACC, BIC, CCV, LCH

Mappe colore: GGE (1 order)

Accuracy: 41.2%

# Todo

* Training su DSO e test su DSI
  + *Accuracy: 41.2%, i due dataset hanno condizioni di luce molto diverse*
* Capire quando i modelli basati su ­­­­­ o gge non sono concordi (capire su quali immagini ha problemi)
  + *In realtà sono sempre concordi: nella votazione a maggioranza o sbagliano entrambi o entrambi danno l’output corretto.*
* Fare le stesse prove su DSI
  + *L’accuratezza è sempre molto alta, con crossvalidazione, oltre 85%*
* Soft values in output
* Due GGE differenti (cambia n, sigma e p) (victor)
* Valutare cambio ordine GGE (secondo metodo)

1. Provare a non segmentare all’interno della singola banda
2. Sliding window
3. Confronto metodi a singola immagine
4. Training su DSI e test su DSO (face detection)
   1. Accuracy: 92%!

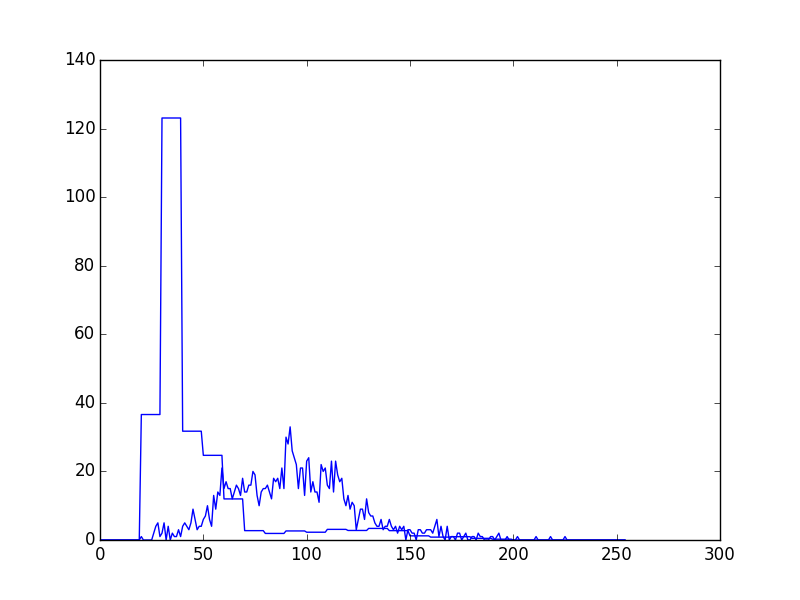
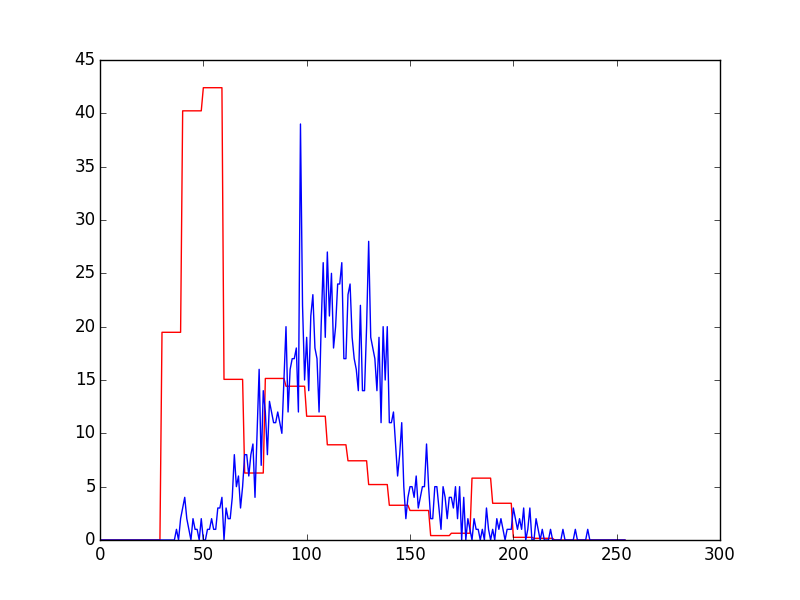
Il training su DSI (molto più variegato), addestra dei modelli migliori.

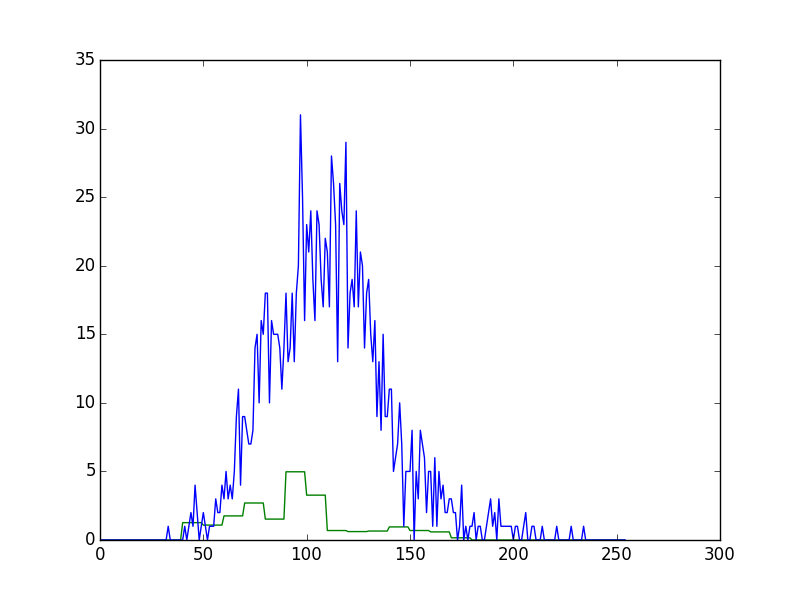
Alcuni esempi (splicing-65, splicing-87): in questi casi, a differenza dei modelli precedenti, l’output è corretto al 100%.

1. Numero di campioni per ciascuna fascia di colore

Nei seguenti grafici, sulle ascisse è rappresentata la media del colore delle bande estratte dall’immagine, sulle ordinate la varianza della differenza del reference color di banda e quello mediano dell’intera immagine, calcolato con 5 algoritmi diversi (la varianza è calcolata su una fascia di valori di 10 unità sulle ascisse).

Dai grafici si nota che, in generale, media del colore delle bande cade tra 75 e 120. Quando ci troviamo in questa situazione, la varianza, specialmente nel canale blu e verde, è molto bassa e tendente allo zero. Questo vuol dire che i 5 algoritmi rispondono tutti allo stesso modo (che assumiamo dunque corretto).





1. Cambiare spazi colore, YCbCr, HSV
2. Utilizzare il metodo a regioni per stabilire quale faccia è tamperata (la colonna e la riga che ingloba la faccia)
3. Test del metodo a regioni su DSO-1
4. Segmentazione post detection per identificare elementi interessanti
5. Feature dell’immagine composta da 5 valori (i 5 valori della GGE)
6. Detector generico, salienza nell’immagine sull’intera immagine (poi intersezione sui blocchi)

# Codice sorgente

Linguaggio utilizzato Python, versione 3.4.

Librerie utilizzate: OpenCV 3.0

Tutto il codice sorgente è completamente aperto e disponibile su <https://github.com/lorecioni/ImageSplicingDetection>.

*Generazione delle Illuminant Maps*

Linguaggio C++

Dipendenze:

- OpenCV (tested with 2.4.0)

- Boost (tested with 1.45)

- cmake (tested with 2.8.2)

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

*Estrazione dei descrittori ACC, BIC, CCV, EOAC, LAS, LCH, SASI, SPYTEC, UNSER*

Linguaggio C++

**Funzione principale: main.py**

usage: main.py [-h] [--face-detector] [--region-detector] [--train] [--detect]

[--output-mask OUTPUT\_MASK] [--display-result]

[--crossvalidate] [--extract-single-features]

[--no-extract-features] [--no-extract-maps] [--img IMG]

[--euclidean-distances] [--heat-map] [--verbose]

optional arguments:

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

--face-detector use face detector

--region-detector use region detector

--train train the model for further splicing detection

--detect detect splice over an image

--output-mask OUTPUT\_MASK

output mask path

--display-result display mask

--crossvalidate 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

--euclidean-distances

visualize distances

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

--verbose display all messages

References:

[1] T. Carvalho, et al. *Illuminant-Based Transformed Spaces for Image Forensics*. IEEE Transactions on Information Forensics and Security 11.4 (2016): 720-733.

[2] C. Riess and E. Angelopoulou, *Scene illumination as an indicator of image manipulation*. In Proceedings of the 12th international conference on Information hiding, Berlin, Heidelberg, 66-80; 2010

[3] V. Schetinger et al. *Exploring Statistical Differences Between Illuminant Estimation Methods for Exposing Digital Forgeries*; 2016.

[4] Yu Fan, Philippe Carré, Christine Fernandez-Maloigne*, Image Splicing Detection With Local Illumination Estimation*, ICIP 2015