UNIVERSITÀ DEGLI STUDI DI FIRENZE

Laurea Magistrale in Ingegneria Informatica

Corso di Analisi di Immagini e Video

Sviluppo ed implementazione di una applicazione basata su Multiscale Local Binary Pattern per il rilevamento di difetti su texture

A. Rizzo, M. Bruni

Anno accademico 2012/2013

Indice

Sommario

In questa relazione, gli autori presentano un'applicazione che determina se regioni di un'immagine contengono malformazioni della tessitura al loro interno. Verrà fatta una panoramica sulle *immagini di tessiture*. Successivamente saranno presentati i concetti teorici alla base dell'operatore *Local Binary Pattern (LBP)*. Verranno descritte delle varianti dell'operatore LBP base, ovvero *Extended LBP*, *Uniform LBP* e *Multi-scale Local Binary Pattern (MLBP)*. Infine verranno mostrati gli strumenti utilizzati per lo sviluppo ed i risultati ottenuti dai test.

1 Texture

Una *texture* è una nozione generale che può avere molteplici significati. In particolare nel contesto della percezione umana una texture è una specifica struttura di superfici, formate da uno o più particolari elementi, che si ripetono nello spazio. Questa ripetizione può riguardare:

- la variazione locale di scala;
- l'orientazione;
- la distribuzione spaziale dei livelli di grigio o del colore[1];
- o altre caratteristiche geometriche dell'elemento.

In figura 1 sono rappresentati alcuni esempi di immagini di tessiture.

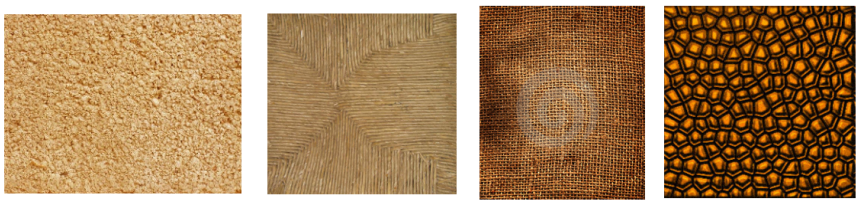

Figura 1: Immagini di tessiture.

Una immagine è percepita contenente tessiture quando il numero di elementi che la compongono è sufficientemente numeroso. Se il numero di elementi presenti nell'immagine è ridotto, allora questi sono percepiti come un gruppo di oggetti numerabili invece che una tessitura[2].

Alcune delle caratteristiche qualitative che possono essere utilizzate per differenziare le texture sono:

- fineness - coarseness,
- smoothness,
- granularity,

- lineation,
- directionality,
- regularity - randomness

L'utilizzo di modelli di tessiture può trovare interessanti applicazioni nel contesto della:

- Segmentazione di immagini;
- Classificazione di immagini.

La *segmentazione* di immagini consiste nel suddividerle in regioni che presentano tessiture uniformi. La *classificazione*, associa ad ogni regione ottenuta dal processo di segmentazione la sua classe di appartenenza.

2 Multi-scale Local Binary Pattern

In questo capitolo verranno presentati alcuni concetti teorici e gli stumenti utilizzati per lo sviluppo dell'applicazione. Il primo paragrafo introdurrà il descrittore di feature *Local Binary Pattern (LBP)*. Verrà analizzata una sua variante, ovvero *Uniform LBP*, che viene utilizzata per ridurre la lunghezza del vettore di *feature*. Analizzeremo infine la sua estensione *Multi-scale Local Binary Pattern (MLBP)*.

2.1 Local Binary Pattern

Local Binary Pattern (LBP) è un efficiente texture operator, introdotto per la prima volta nel 1994 [3]. Le feature estratte con questo metodo vengono utilizzate per la classificazione di immagini in ambito della computer vision.

LBP associa ai pixel dell'immagine un'etichetta, determinata tramite un'operazione di confronto tra il pixel in esame ed un certo numero di pixel nelle sue vicinanze, *neighborhood*. Tra le caratteristiche principali di LBP troviamo:

- è robusto alla variazione uniforme dell'illuminazione dell'immagine;
- è robusto al rumore;
- non è invariante alle rotazioni dell'immagine;
- non è invariante alla scala.

Un altra importante proprietà è infine la sua semplicità computazionale che lo rende ideale per l'analisi delle immagine in Real-Time (RT).

2.1.1 Descrizione

L'idea alla base dell'operatore Local Binary Pattern (LBP) è che le texture bidimensionali possono essere descritte attraverso pattern locali. L'implementazione base dell'operatore LBP, etichetta i pixel dell'immagine attraverso un'operazione di sogliatura dei pixel contenuti in una maschera 3 x 3.

Il pixel centrale viene confrontato con ogni vicino ed il risultato del confronto viene espresso con un numero binario.

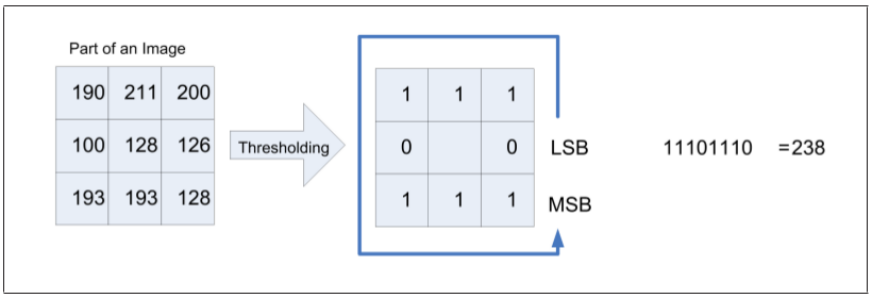

Figura 2: Estrazione del codice LBP.

Riferendoci alla figura 2 il processo di etichettatura dei pixel è il seguente: il livello di grigio del pixel centrale, detto anche *pivot* viene confrontato con il livello di grigio di tutti gli altri pixel della maschera che viene binarizzata attraverso la seguente funzione:

$$s(x) = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{se } x < 0 \end{cases} \tag{1}$$

dove x è la differenza tra l'intensità del livello di grigio di un pixel del *neighborhood* ed il livello di grigio del pixel pivot. I valori così ottenuti vengono concatenati in senso antiorario andando a formare un codice binario di 8 bit la cui conversione decimale rappresenta il codice LBP del pixel pivot.

Descrittore

Il numero totale di codici LBP è pari a $2^8 = 256$. L'istogramma dei codici LBP calcolati sui pixel dell'immagine può essere utilizzato come descrittore della texture. In figura 3 è mostrato l'istogramma di un immagine LBP.

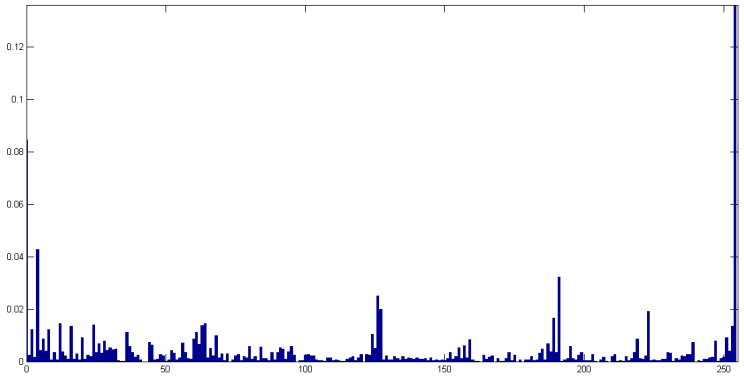

Figura 3: Istogramma normalizzato di un'immagine LBP.

Per il calcolo dell'istogramma si utilizza la seguente formula:

$$h(i) = \sum_{x,y} B(LBP_{P,R}(x,y) = i), \quad i \in [0, 2^P - 1] \tag{2}$$

dove

$$B(v) = \begin{cases} 1, & \text{se v = true} \\ 0, & \text{altrimenti} \end{cases} \tag{3}$$

La formula 2, conta il numero di pixel che appartengono ad ogni rettangolo dell'istogramma.

2.2 Extended LBP

Finora abbiamo considerato come neighborhood i pixel adiacenti a quello di pivot. Una variante dell'operatore LBP è *Extended LBP (ELBP)* o Circular LBP (CLBP). ELBP considera come neighborhood i P pixel che si trovano ad una distanza R dal pivot. Le coordinate dei P pixel si possono ottenere con la

seguente formula:

$$(x_p, y_p) = \left(-Rsin\left(\frac{2\pi p}{P}\right), Rcos\left(\frac{2\pi p}{P}\right)\right), \quad p = 0, \cdots, P-1 \tag{4}$$

Nel caso in cui le coordinate ottenute non corrispondono alla griglia discreta dell'immagine si effettua una interpolazione bilineare. In figura 4 sono mostrati alcuni risultati dell'applicazione della formula appena descritta.

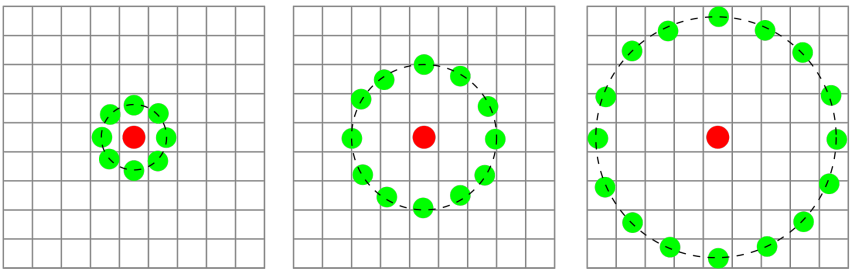

Figura 4: Neighborhood per il calcolo del codice LBP nei casi: $(P = 8, R = 1)$, $(P = 12, R = 2)$ e $(P = 16, R = 4)$.

La formula seguente permette di calcolare in base dieci, il codice LBP di lunghezza P e raggio R generico di un pixel.

$$LBP_{P,R}(x, y) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{5}$$

dove g_c è il pixel centrale, g_p i pixel appartenenti al neighborhood e la funzione $s(x)$ è la funzione definita in Eq.1.

2.3 Uniform Local Binary Pattern

Come accennato precedentemente, l'istogramma dell'immagine LBP è utilizzato come descrittore della texture. Utilizzando l'operatore LBP base, otteniamo un istogramma con 256 classi. Per ridurre la dimensione del descrittore si può utilizzare *Uniform LBP*. Uniform Local Binary Pattern considera solo i pattern

in cui occorrono al più due transizioni da 0 a 1 o da 1 a 0 tra pixel adiacenti. Per esempio:

- 00000000 → 0 transizioni
- 01110000 → 2 transizioni
- 11001111 → 2 transizioni
- 11001001 → 4 transizioni
- 01010010 → 6 transizioni

I pattern che soddisfano le condizioni di Uniform LBP sono in totale 58, come mostrato in figura 5.

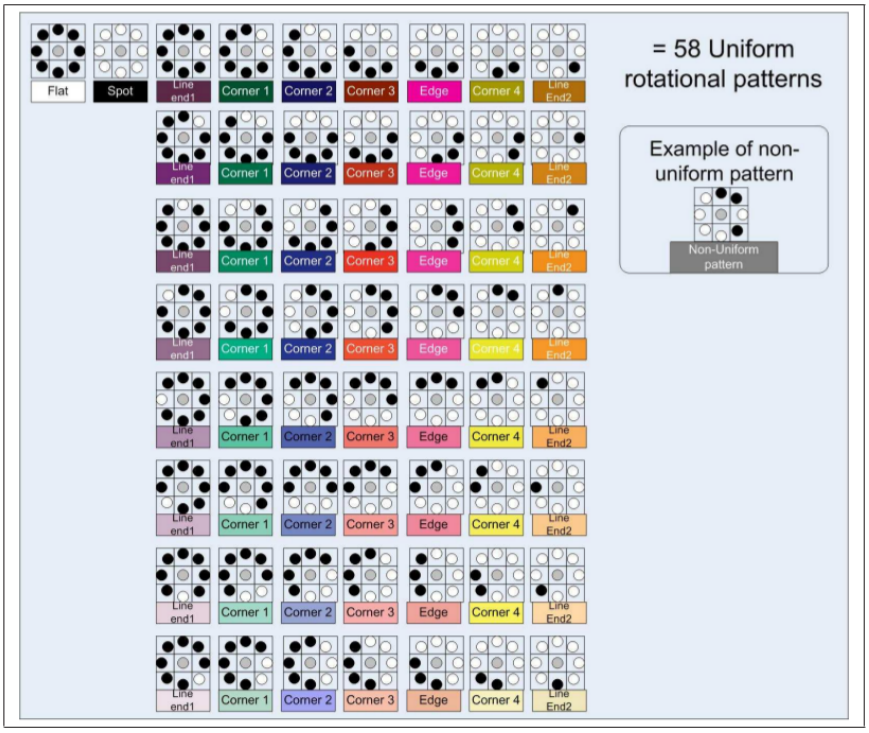

Figura 5: Pattern di Uniform LBP.

Questa variante si basa sul fatto che le feature così estratte sono più robuste al rumore e l'istogramma risultante è di dimensione inferiore mantenendo comunque le feature rilevanti.
In figura 6 è mostrato l'istogramma di un immagine su cui è stato applicato l'operatore di Uniform LBP.

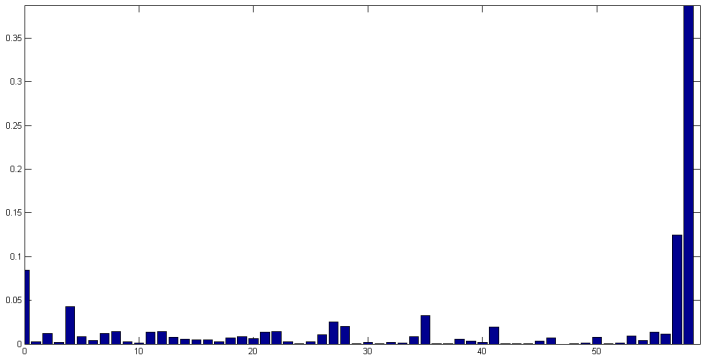

Figura 6: Istogramma normalizzato di una immagine su cui è stato applicato l'operatore di Uniform LBP. I primi 58 pattern sono quelli uniformi, mentre l'ultimo pattern corrisponde a tutti i pixel con codice non uniforme.

Per ottenere il codice di Uniform LBP si utilizza la formula seguente:

$$LBP^{u2}_{P,R}(x,y) = \begin{cases} I(LBP_{P,R}(x,y)), & \text{se } U(LBP_{P,R}) \leq 2, I(z) \in [0,(P-1)P+2) \\ (P-1)P+2, & \text{altrimenti} \end{cases} \tag{6}$$

dove la funzione $U(x)$ calcola il numero di transizioni.

$$U(LBP_{P,R}) = |s(g_{P-1} - g_c) - s(g_0 - g_c)| + \sum_{p=1}^{P} |s(g_p - g_c) - s(g_{P-1} - g_c)| \tag{7}$$

2.4 Multiscale Local Binary Pattern

Multi-scale Local Binary Pattern (MLBP) è un'estensione dell'operatore LBP. MLBP è ottenuto combinando i descrittori delle texture ottenute facendo variare il raggio $r \in \{r_1, r_2, \cdots, r_R\}$ per la determinazione dei neighbor. Alternativamente MLBP può essere ottenuto applicando iterativamente l'operatore LBP con raggio costante ma riducendo ad ogni passo la risoluzione dell'immagine. Quest'ultimo metodo è meno efficace in quanto riducendo la risoluzione dell'immagine risulta più difficile estrarre informazioni sul contrasto tra piccole regioni lontane fra loro.

Le feature estratte con MLBP risultano migliori rispetto a LBP per la classificazione di immagini[4].

2.4.1 Descrittore

Il descrittore della texture ottenuto con MLBP è la concatenazione degli istogrammi ottenuti applicando iterativamente LBP sulla stessa immagine al variare del raggio $r \in \{r_1, r_2, \cdots, r_R\}$. Il descrittore è dato dalla seguente formula:

$$f = [h_{P,r_1}, h_{P,r_2}, \cdots, h_{P,r_R}] \tag{8}$$

dove

$$h_{P,r}(i) = \sum_{x,y} B(LBP_{P,r}(x,y) = i), \quad i \in [0, L-1], r \in \{r_1, r_2, \cdots, r_R\} \tag{9}$$

con L numero massimo di classi dell'istogramma e

$$B(v) = \begin{cases} 1, & \text{v = true} \\ 0, & \text{altrimenti} \end{cases} \tag{10}$$

2.5 Filtro di smoothing gaussiano

I filtri di *smoothing* vengono utilizzati principalmente per ridurre il rumore presente nell'immagine. Sono anche detti *filtri di media*. Infatti facendo scorrere lungo l'immagine la maschera del filtro, detta anche *kernel mask*, il valore di ogni pixel viene sostituito con la media pesata dei livelli di grigio dei pixel interni alla regione della maschera. Il processo di smoothing permette di ridurre i dettagli meno significativi dell'immagine e mettere in risalto le caratteristiche strutturali della stessa. Un effetto indesiderato dello smoothing è quello di produrre un'immagine sfocata, soprattutto se applicato iterativamente. La figura 7 mostra un filtro di smoothing generico 3x3. La maschera del filtro verrà moltiplicata per un coefficiente di normalizzazione in modo tale che la somma degli elementi della maschera faccia uno. I coefficienti della maschera vengono scelti secondo il seguente principio: il peso associato al pixel centrale assume un valore superiore rispetto agli altri. I pesi associati agli altri pixel assumono valori decrescenti all'aumentare della distanza dal pixel centrale. In altre parole si vuole fare in modo che il pixel centrale abbia una maggiore importanza nel calcolo della media. Viceversa i pixel più lontani da quello centrale peseranno di meno nel calcolo della media. Questo permette di ridurre l'effetto indesiderato del processo di smoothing.

$\omega_{1,1}$	$\omega_{1,2}$	$\omega_{1,3}$
$\omega_{2,1}$	$\omega_{2,2}$	$\omega_{2,3}$
$\omega_{3,1}$	$\omega_{3,2}$	$\omega_{3,3}$

Figura 7: Esempio di una kernel mask di dimensioni 3x3, dove $\omega_{i,j}$ reppresenta il peso applicato al valore del livello di grigio del corrispettivo pixel nell'immagine.

I coefficienti della maschera sono calcolati utilizzando la *2D Gaussian Smoothing Operator* $G(x, y)$, cui formula è riportata qui di seguito:

$$G(x, y) = e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{11}$$

La dimensione n della kernel mask è legata al valore di σ dalla seguente formula:

$$6\sigma - 1 = n \tag{12}$$

3 Implementazione

In questo capitolo verranno illustrati gli strumenti e gli algoritmi utilizzati per l'implementazione del software.

3.1 Strumenti

Per lo studio e l'implementazione del Multi-scale Local Binary Pattern (MLBP) è stata creata una libreria in Matlab. Matlab (MATrix LABoratory) è un ambiente per il calcolo numerico e l'analisi statistica che comprende anche l'omonimo linguaggio di programmazione creato dalla MathWorks [5].

La libreria creata è stata utilizzata in un'applicazione per la rilevazione di errori all'interno di immagini di tessiture. Per permettere un uso agevole del software e della configurazione dei parametri è stata sviluppata una Graphical User Interface (GUI), mostrata in figura 8. I dati inseriti all'interno dei campi di input devono rispettare la sintassi del linguaggio Matlab. Ad esempio per i campi *radii* e *training set*, in cui è possibile inserire un array di valori, è accettata la seguente sintassi:

```
inizio:step:fine [, inizio2:step2:fine2]
```

ad esempio se si vuole ottenere un array con i valori: $1, 3, 5, 7$, basterà scrivere: $1 : 2 : 7$.

Il codice sorgete di tutta l'applicazione è disponibile sul servizio di versioning Github (http://dining-engineers.github.io/Multi-scale-Local-Binary-Pattern/).

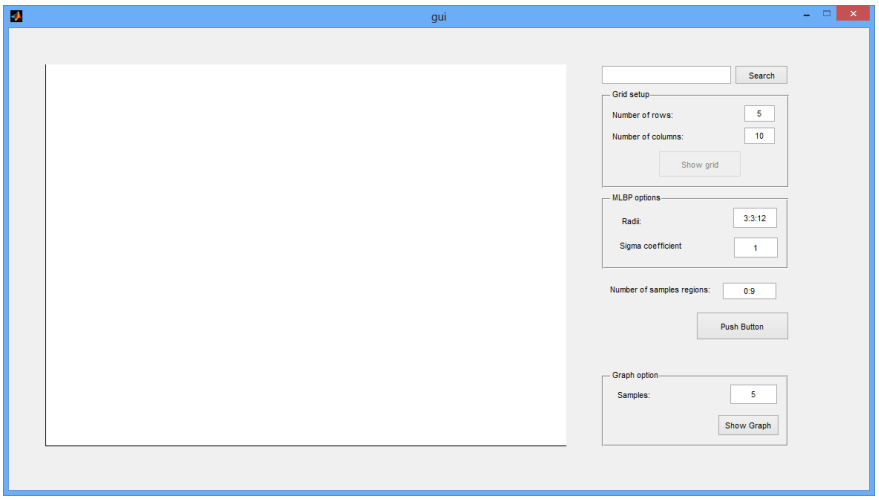

Figura 8: GUI.

Le directory sono organizzate nel modo seguente:

- *Dataset*: contiene le immagini del dataset;
- *Documents*: contiene tutti i documenti consultati per lo sviluppo dell'applicazione. In particolare, nella sottodirectory *relazione* sono contenuti i sorgenti LaTeX della relazione del il file PDF della stessa;
- *src*: contiene il codice sorgente dell'applicazione.

L'organizzazione delle directory è raffigurata nella figura 9

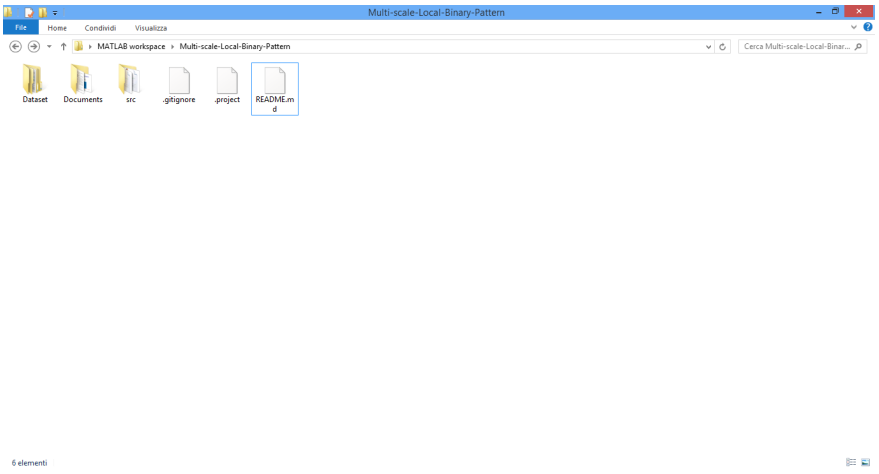

Figura 9: Organizzazione delle directory dell'applicazione.

3.2 Analisi dell'immagine

Nel contesto della nostra applicazione, le immagini in esame saranno analizzate in scala di grigio. Per ottenere un descrittore della texture dell'immagine è stata implementata la funzione

```
function [ descriptor ] = getMLBPDescriptor( img,
    mapping, radii, num_region_rows, num_region_cols,
    sigma_coefficient )
```

che richiede in input: l'immagine da elaborare, la variante di LBP da utilizzare (nel nostro caso Uniform LBP), un array contenente i raggi per fare MLBP, il numero di righe e colonne in cui l'immagine verrà suddivisa (paragrafo 3.2.1) ed infine il valore della deviazione standard per l'applicazione del filtro di smoothing gaussiano.

Al suo interno, getMLBPDescriptor(), applica iterativamente all'immagine l'operatore LBP. Ad ogni iterazione i, all'immagine viene inizialmente applicato il filtro di smoothing gaussiano

```
% apply gauss smoothing
myfilter = fspecial( 'gaussian', kernel_size,
    sigma_coefficient );
```

```
img = imfilter( img, myfilter, 'replicate' );
```

Successivamente viene chiamata la funzione[6]

```
function [lbp_img] = lbp( img, radius, num_neighbors,
    mapping );
```

che calcola l'immagine LBP con raggio $radii(i)$, (paragrafo 2.2).

3.2.1 Suddivisione dell'immagine in regioni

Al fine di determinare le regioni della tessitura in cui sono presenti degli errori, abbiamo deciso di suddividere l'immagine in regioni non sovrapposte. I parametri num_region_rows e num_region_cols determinano il numero di regioni in cui l'immagine viene suddivisa. Le regioni sono numerate da 0 a $num\_region\_rows \cdot num\_region\_cols - 1$. Le coordinate della k-esima regione sono ottenute attraverso l'utilizzo della funzione

```
function [ rMin, rMax, cMin, cMax ] = gridBounds(
    imgSize, num_region_rows, num_region_cols, k )
```

In figura 10 viene mostrata la suddivisione in regioni dell'immagine di input.

In seguito all'operazione di segmentazione dell'immagine, per ogni regione, viene estratto il relativo descrittore MLBP:

$$f = [h_{P,r_1}, h_{P,r_2}, \cdots, h_{P,r_R}]$$

come spiegato nel paragrafo 2.4.1.

3.3 Misura di similarità

Per misurare la similarità tra i descrittori di due regioni I e J, possono essere utilizzati vari criteri. Abbiamo implementato le seguenti misure di similarità $Sim(I, J)$:

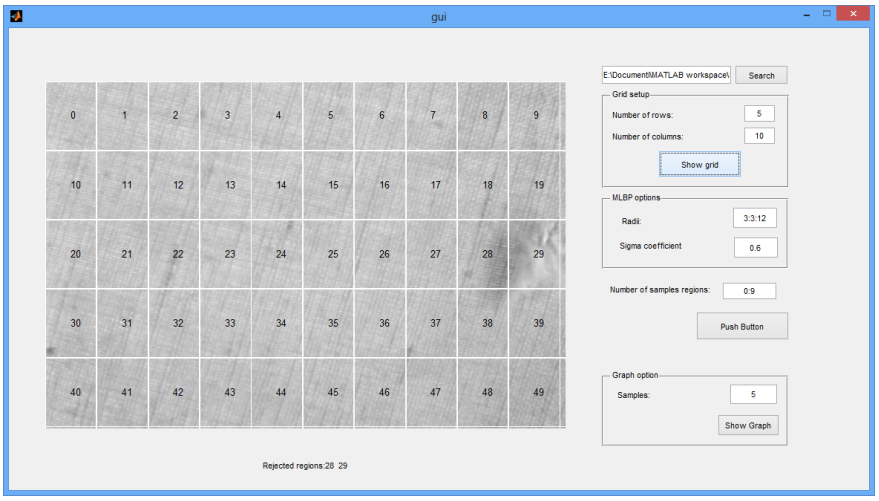

Figura 10: Screenshot della GUI in cui viene mostrata la suddivisione in regioni dell'immagine di input.

- Chi-square criterion:

$$Sim(I,J) = -\sum_i \frac{(f_I(i) - f_J(i))^2}{f_I(i) + f_J(i)} \tag{13}$$

- Histogram intersection:

$$Sim(I,J) = \sum_i \min(f_I(i), f_J(i)) \tag{14}$$

- Log-likelihood ratio (Kullack-Leibler divergence):

$$Sim(I,J) = -\sum_i f_I(i) log(f_J(i) \tag{15}$$

- Normalize Correlation:

$$Sim(I,J) = \frac{f_I f'_J}{||f_I||||f_J||} \tag{16}$$

- Normalize Cross-correlation:

$$Sim(I, J) = \frac{\sigma_{f_I f_J}}{\sigma_{f_I} \sigma_{f_J}} \tag{17}$$

Tra le misure sopra descritte, è stato scelto di utilizzare la misura di similarità *Normalize Correlation*. Nel caso in cui l'utente preferisca utilizzare un'altra misura di similarità è sufficiente modificare il codice nella funzione *computeRejectedRegions*, dove viene specificata la misura di similarità in uso.

3.4 Testing

Lo scopo dell'applicazione è quello di determinare se alcune regioni presentano malformazioni della tessitura al loro interno. Per poter classificare una regione come corretta o errata, è necessario conoscere un certo numero di regioni sicuramente corrette. Come possiamo vedere in figura 12, l'utente può specificare nel campo *number of samples regions* il numero delle regioni esatte che vanno a definire il *training set*.

Ogni regione corretta viene confrontata con tutte le altre regioni del training set utilizzando la misura di similarità $Sim(I, J)$, dove I e J rappresentano i descrittori MLBP. In questo modo si ottengono le misure di similarità del training set da cui possiamo estrarre un valore medio ($\mu_{training}$) ed una deviazione standard ($\sigma_{training}$) che definiscono una distribuzione gaussiana delle misure di similarità delle regioni corrette.

Per classificare una regione k, che non appartiene al training set, si confronta il suo descrittore con tutti i descrittori del training set utilizzando la medesima misura di similarità $Sim(I, J)$. In questo modo si ottengono le misure di similarità della regione di test con il training set da cui possiamo estrarre un valore medio (μ_{test}) ed una deviazione standard (σ_{test}).

La regione k verrà classificata come corretta se rispetta la seguente condizione:

$$|\mu_{training} - \mu_{test}| \leq 3\sigma_{training} \tag{18}$$

altrimenti verrà classificata come una regione contenente difetti.

3.4.1 Dataset

Il *dataset* utilizzato per testare l'applicazione consiste in un insieme di immagini che raffigurano la trama di tessuti. All'interno di alcune di queste immagini sono presenti uno o più difetti che la procedura dovrà identificare. In figura 11 sono rappresentate alcune immagini del dataset.

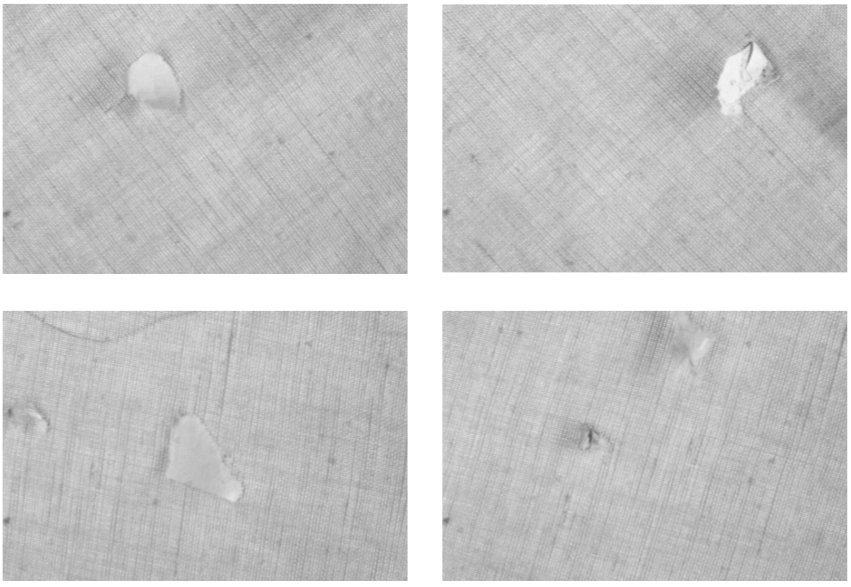

Figura 11: Esempio di immagini utilizzate nel dataset.

3.5 Risultati e conclusioni

Applicando la regola di classificazione descritta nel paragrafo precedente, ogni regione viene classificata ed i risultati vengono riportati direttamente sull'immagine come mostrato nella figura 12.

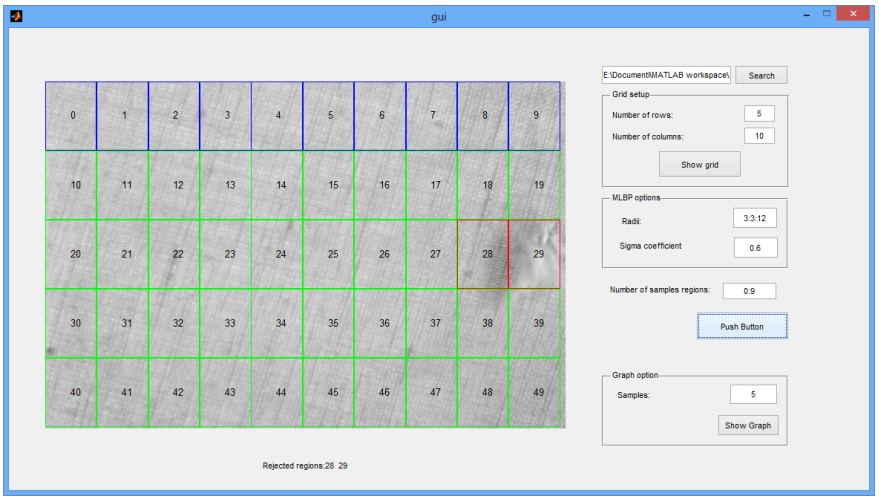

Figura 12: Risultati dell'operazione di classificazione. Le regioni con contorno di colore rosso sono quelle contenenti difetti, le regioni di colore blu sono quelle del training set e le regioni con contorno verde sono quelle ritenute corrette.

L'utente, cliccando sull'apposito bottone *Show Graph*, può visualizzare i grafici delle distribuzioni gaussiane determinate precedentemente. In figura 13 è visualizzato un esempio dei grafici delle distribuzioni gaussiane delle cinque regioni con media μ_{test} peggiori.

In figura 14, è visualizzato un esempio dei grafici delle distribuzioni gaussiane delle cinque regioni con media μ_{test} peggiori tra quelle classificate come regioni corrette.

I risultati ottenuti in seguito all'applicazione del metodo da noi implementato su un numero significativo di immagini, si sono rivelati soddisfacenti. Infatti nei casi di classificazione di falsi positivi e veri negativi è bastato modificare opportunamente i parametri di configurazione per ottenere una classificazione esatta.

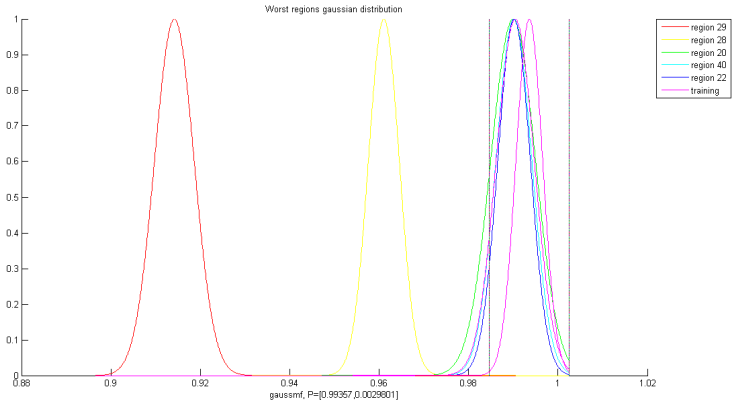

Figura 13: Caso peggiore

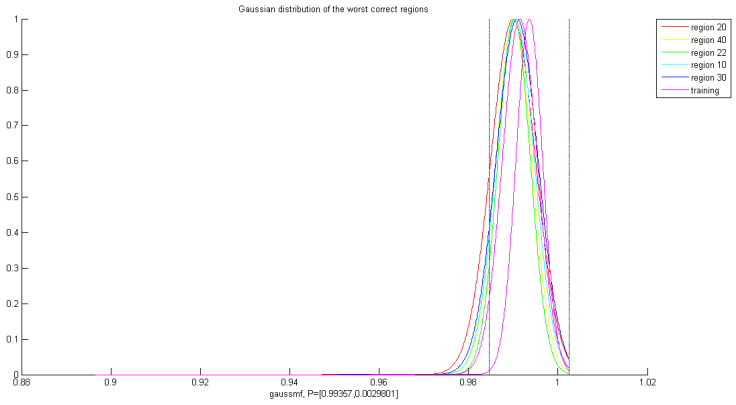

Figura 14: I peggiori tra i migliori

Acronimi

MLBP Multi-scale Local Binary Pattern

LBP Local Binary Pattern

RT Real-Time

ELBP Extended LBP

CLBP Circular LBP

GUI Graphical User Interface

Elenco delle figure

Riferimenti bibliografici

[1] G. Stockman and L. G. Shapiro, *Computer Vision*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[2] 3.02 - local feature - texture description.

[3] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision amp; Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1, 1994, pp. 582–585 vol.1.

[4] C. H. Chan, "Multi-scale local binary pattern histogram for face recognition."

[5] MATLAB, *version 8 (R2013a)*. Natick, Massachusetts: The MathWorks Inc., 2013.

[6] M. Heikkila and T. Ahonen. Local binary pattern (lbp) implementation for matlab. [Online]. Available: http://www.cse.oulu.fi/CMV/Downloads/LBPMatlab