Capitolo 3

Analisi e Sintesi di Tessiture: *Clustering* e Multirisoluzione

In questo capitolo descriviamo un nuovo metodo per l'analisi e sintesi di tessiture che fa uso di un approccio multirisoluzione. Nella fase di analisi tale tecnica realizza una operazione di *clustering* volta a compattare insieme le *feature* strutturali percettivamente indistinguibili apprese da una tessitura di *input*, facendo uso di un'opportuna tecnica di decomposizione a piramide. Il processo di sintesi è ottenuto attraverso una procedura di campionamento che vincola opportunamente la distribuzione delle caratteristiche spaziali ai vari livelli di risoluzione. A tal fine si fa uso di un a particolare struttura dati detta *frequency*

tree. Alcuni risultati sperimentali mostrano come effettivamente tale approccio permetta di riprodurre il processso di generazione stocastica di una vasta classe di immagini rappresentanti tessiture reali. In particolare il metodo qui proposto risulta essere computazionalmente più efficiente degli analoghi approcci presenti in letteratura.

1. Introduzione

L'algoritmo presentato in questo capitolo, analogamente a De Bonet (1997), fa uso di tecniche di decomposizione a piramide, che come noto evidenziano i dettagli di un'immagine a diversi livelli di risoluzione. Si è però riusciti ad ottenere un notevole *speed-up* nella fase di sintesi della tessitura di *input*, grazie ad un opportuna tecnica di analisi e di elaborazione di *cluster*, attuata ai diversi livelli della piramide. Analoghe tecniche sono state sviluppate in contesti del tutto differenti da Lepsoy e Oien (1994) e Ramamurthi e Gersho (1986).

La clusterizzazione permette di raggruppare insieme le regioni aventi caratteristiche strutturali simili rispetto ad un determinato *detector* di *feature*.

La sintesi vera e propria di nuove tessiture è quindi ottenuta campionando/sorteggiando dalla distribuzione costruita nella fase di analisi. Tale distribuzione è vincolata dalla storia relativa delle frequenze dei vari *cluster* ai vari livelli (rimandiamo ai paragrafi successivi per maggiori dettagli). Il principale aspetto originale della tecnica di sintesi presentata è quindi legata sia alla migliore efficienza computazionale sia alla qualità visiva della tessiture sintetizzate. Risultati sperimentali mostrano infatti come tale algoritmo sia nettamente più efficiente dell'algoritmo di De Bonet.

2. Il *Clustering*

L'algoritmo, di cui discuteremo i dettagli nel paragrafo successivo, partiziona i *pixel* di ciascun livello della piramide Gaussiana, utilizzando un processo algoritmico detto di *clustering*. Mediante il *clustering* è possibile ottenere, un certo numero *k* di sottoinsiemi disgiunti dell'insieme di partenza, grazie ad un'opportuna misura di verosimiglianza o di similarità. Tale operazione può quindi essere utilizzata ogni qualvolta è interessante classificare specie, riassumere risultati o ancora analizzare immagini satellitari. Di conseguenza il *clustering* rappresenta un operazione spesso essenziale, nell'analisi statistico-sperimentale dei dati ed ha assunto un ruolo centrale anche negli algoritmi di visione (Chi, Yan, Pham - 1996).

Nell'algoritmo qui presentato si è utilizzata come misura di similarità una semplice isotropica distanza euclidea nello spazio delle *feature*. Si potrebbero ottenere migliori risultati lavorando con metriche più vicine alla percezione umana, ma questo inevitabilmente introdurrebbe un costo computazionale più alto. Esistono svariati algoritmi di clustering: *c-mean clustering(hard/fuzzy), adaptive vector quantization (AVQ), self-organizing map (SOM)* (Chi et al. 1996; Dickerson and Kosko 1993; Kohonen 1990).

Nei nostri esperimenti abbiamo utilizzato una semplice procedura di *hard c-mean clustering*: iterativamente si creano le classi della partizione attorno ai cosiddetti *centroidi*, punti virtuali dello spazio *n*-dimensionale in cui si lavora, opportunamente stimati ed individuati dall'algoritmo stesso. Inizialmente si era adottato un *fuzzy c-mean clustering*. L'idea infatti di valutare con un'opportuna funzione di *membership*, l'appartenenza di ogni *pixel* alle varie classi della

partizione sembrava dare più garanzie di versatilità. In pratica però ci si è accorti che al maggiore sforzo computazionale non corrispondeva un sensibile *improvement* nella qualità delle tessiture sintetizzate.

3. Il Nostro Approccio

Abbiamo già discusso in maniera dettagliata dell'algoritmo di De Bonet (1997) analizzandone i dettagli implementativi nei suoi vari aspetti. La tecnica algoritmica qui proposta realizza fondamentalmente due *improvement* rispetto a tale algoritmo. Il primo è strettamente legato all'efficienza computazionale; in particolare il nostro metodo è computazionalmente più veloce sia della versione base dell'algoritmo di De Bonet sia di quella migliorata con una serie di euristiche implementative suggerite dallo stesso autore. L'idea originale che sta alla base di questo risultato è legato alla scelta di ridurre lo spazio di ricerca e/o di campionamento attraverso una opportuna procedura di *clustering* realizzata ai vari livelli di risoluzione della piramide. Il secondo significativo risultato viene fuori direttamente dall'euristica seguita nella realizzazione del *clustering*: si ottiene infatti un esplicito modello statistico della tessitura di *input*. Questo modello, detto *frequency tree*, è descritto appunto da una struttura dati di tipo albero e contiene informazioni esplicite circa la co-occorrenza delle *feature* sotto osservazione rispetto ai vari livelli di risoluzione. Si osservi che tale struttura nel nostro approccio è costruita solamente una volta, per ciascuna tessitura considerata. Ciò significa che successive generazioni casuali della stessa tessitura non richiedono nessuno sforzo computazionale aggiuntivo, a differenza invece di ciò che accade nell'algoritmo di De Bonet. In ogni caso l'idea di base del

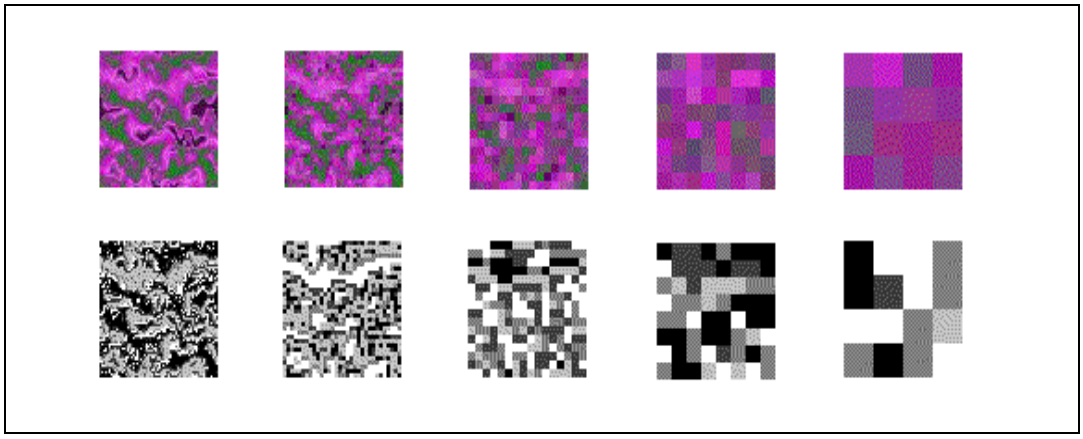

Fig. 1. Una piramide Gaussiana standard con relativa clusterizzazione

frequency tree va comunque oltre la sintesi di *pattern/texture* visuali; ci si aspetta infatti di poter utilizzare tale approccio per poter identificare o comunque prendere traccia della tessitura sotto osservazione trovandone quindi applicazione diretta nel campo dell'*image retrieval* o della discriminazione di tessiture. L'algoritmo opera distinguendo due fasi. Nella prima l'immagine in *input* viene analizzata rispetto alle sue componenti strutturali a diversi livelli di risoluzione. Sulla base delle informazioni raccolte è quindi possibile campionare opportunamente la tessitura in oggetto e sintetizzarla di conseguenza. I dettagli sono riportati nei successivi due paragrafi.

3.1 Analisi

La tessitura di *input* ***I*** è decomposta attraverso una procedura *standard* di decomposizione a piramide. Tale operazione permette di caratterizzare i dettagli peculiari della particolare tessitura che si sta analizzando, ai vari livelli di risoluzione. Denotiamo con $G_0 = \boldsymbol{I}$, G_1 , … ,G_N e rispettivamente con L_1 , … , L_{N-1}

i corrispondenti "spazi delle ottave" (*octave-spaces*) della piramide Gaussiana e della piramide Laplaciana.

Ciascuna ottava Gaussiana è quindi convoluta con un insieme di filtri al fine di evidenziarne le caratteristiche salienti, che costituiranno poi le *feature* sulle quali lavorerà l'algoritmo. Analogamente a quanto realizzato da De Bonet si è deciso di utilizzare semplici filtri di Sobel e un binomiale 3x3. Utilizzando le informazioni ottenute da tali operazioni, ciascun livello della piramide Gaussiana viene quindi *clusterizzato*. E' in questo modo possibile riuscire a suddividere l'insieme dei *pixel* in una serie di sottoinsiemi o di sottogruppi legati tra loro da una qualche misura di dissimilarità. Nel nostro caso si è adottata una semplice procedura di *hard c-mean*. La Figura 1 mostra, evidenziandoli con differenti toni di grigio, le partizioni e quindi i *cluster*, ottenuti ai vari livelli di risoluzione di una particolare *texture*. Il numero di *cluster* da considerare è ovviamente un parametro cruciale, soprattutto se riferito alla qualità dei risultati. Si è comunque sperimentalmente accertato che alle alte e medie risoluzioni cinque *cluster* sono un ottima scelta mentre ai livelli inferiori si potrebbe fare uso di un numero inferiore di *cluster*. Nel nostro caso comunque si è deciso di trattare i vari livelli di risoluzione in maniera equanime senza peraltro risentirne dal punto di vista strettamente computazionale. Denotiamo con $\mathbf{P}_{ij}$, $i=0, \ldots, N-1$, $j=1, \ldots , k_i$ il j-esimo cluster ottenuto all i-esimo livello di risoluzione. Ne segue che al livello i si avranno k_i *cluster* differenti. La conoscenza specifica delle classi $\mathbf{P}_{ij}$ permette la generazione di una storia legata a ciascun *pixel* ed ai suoi immediati vicini lungo i vari livelli della piramide. In particolare è possibile calcolare la frequenza relativa di ciascun *cluster* al livello i rispetto ai *cluster* del livello immediatamente inferiore relativi

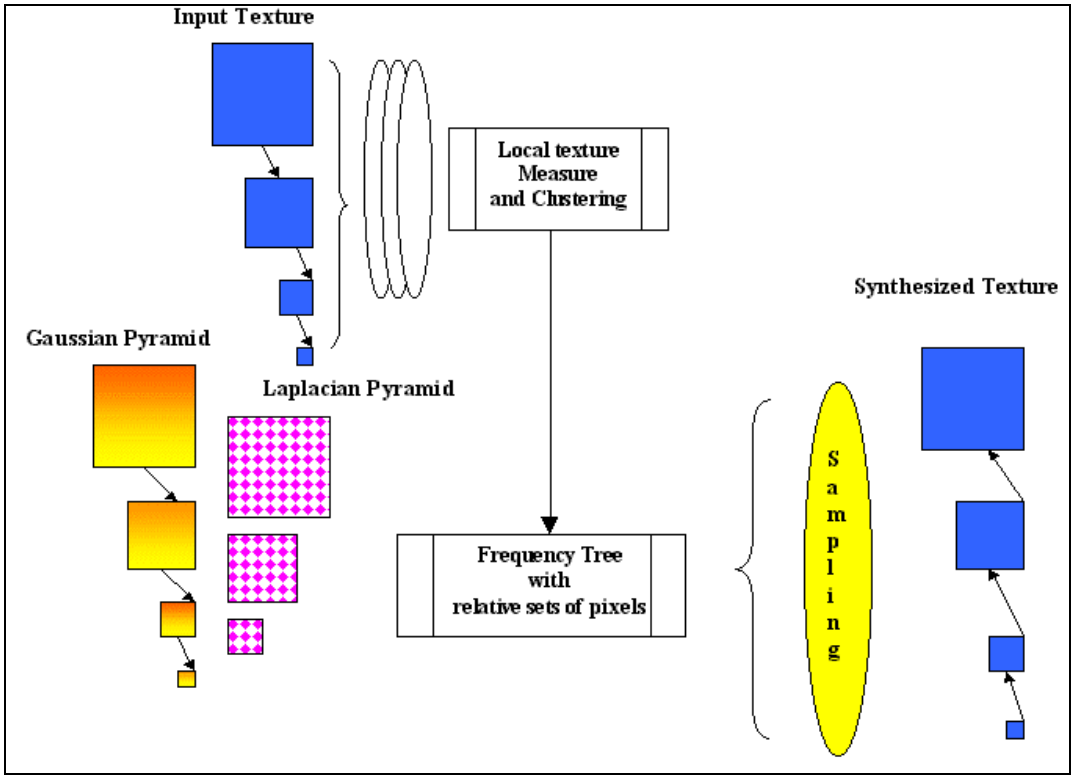

Fig. 2. Una descrizione schematica dell'algoritmo.

al livello di risoluzione più basso *i+1*. Le informazioni collezionate in questa fase vengono organicamente inserite in un albero che riassume le proprietà statistiche della popolazione di *pixel* che fa capo alla tessitura in oggetto. Questo albero viene indicato con il nome di *frequency tree* della tessitura di *input*. Tale albero avrà altezza pari al numero dei livelli di risoluzione considerati, mentre il numero di possibili "cammini" lungo l'albero sarà dato da k_1 x k_2 x .. x k_{N-1}.

A ciascuna foglia corrisponde un insieme di *pixel*: tutti i *pixel* nella stessa foglia condividono la stessa *storia*, nel senso da noi appena considerato, ai diversi livelli di risoluzione.

3.2 Sintesi

La sintesi di una nuova tessitura si basa su un esplicito modello statistico: il *frequency tree* costruito nella fase di analisi.

Phase 0

Ricostruzione del livello di risoluzione più basso.

Phase 1

Per ogni successivo livello *i*:

- Assegna a tutti i *pixel* di questo livello uno dei cluster $\mathbf{P}_{i1},\ldots\mathbf{P}_{ik_i}$ tenendo conto delle frequenze relative presenti nel *frequency tree*. In questo modo si costruisce una completa storia fino al livello *i*-esimo per ciascuno dei *pixel*. Si osservi che i *pixel* di un presenti all'interno di un quadrato 2x2 condividono la stessa storia fino al livello (*i-1*)-esimo. Solo a questo punto è consentito loro di differenziarsi in accordo con le informazioni memorizzate nella fase di analisi. Ciascun quadrato 2x2 del livello *i*-esimo, viene quindi caratterizzato dalla stessa storia fino al livello *i-1*, e da quattro etichette relative ai *cluster* assegnati a ciascun pixel al livello *i*. Denotiamo la storia di ciascun quadrato 2x2 *Q* con $h_Q = (h, j_1, j_2, j_3, j_4)$.
- Per ogni intorno 2x2 *Q* in questo livello di risoluzione assegna un quartetto 2x2 i cui elementi siano terne RGB, prese dalla piramide Laplaciana della tessitura di input, al corrispondente livello di risoluzione, che abbiano lo stesso vettore storico h_Q.

Phase 2

Attraverso un'operazione standard di *collapse* si ricostruisce la tessitura.

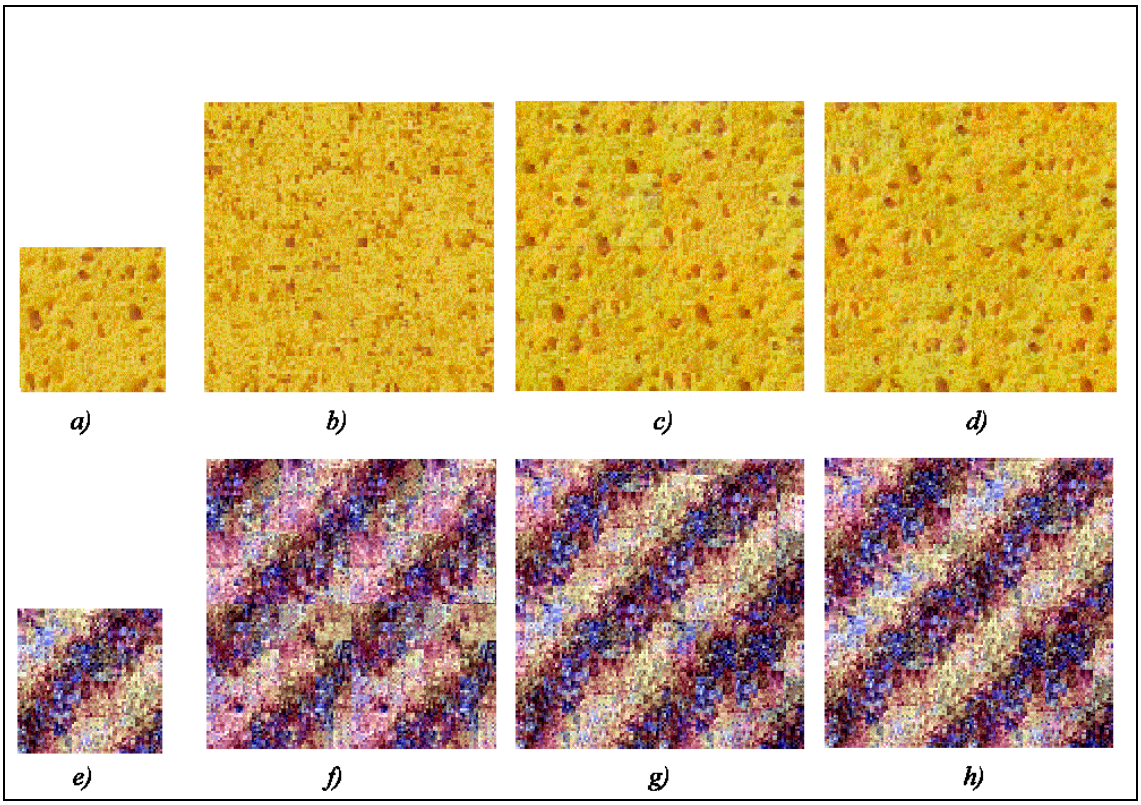

Fig 3. Risultati sperimentali

Tale processo algoritmico è brevemente schematizzato in Fig. 2. Per ogni posizione ad ogni livello si realizzano due semplici campionamenti *random* vincolati: uno per assegnare ad ogni *pixel* un dato *cluster*, l'altro per assegnare il corrispondente quartetto di valori RGB Laplaciani. Data una tessitura 2^N x 2^N , se l'analisi viene eseguita su N livelli di risoluzione, l'algoritmo proposto richiede $O(2^{2N})$ operazioni di campionamento *random*. Questo *upper bound* non può invece essere garantito dall'algoritmo di De Bonet (1997) dove il campionamento deve necessariamente essere eseguito ogni volta sull'intera popolazione dei *pixel*. Ciò significa che bisogna visitare per ciascuna *pixel* da campionare, tutti i corrispondenti *parent vector* relativi a tutti i *pixel* della corrispondente piramide di

input, e sorteggiare fra quelli trovati (si veda il capitolo precedente per maggiori dettagli). Il vantaggio computazionale del nostro approccio è già evidente nel sintetizzare tessiture della stessa dimensione dell'originale. Si accentua se invece consideriamo la possibilità di ottenere tessiture di dimensioni maggiori rispetto a quelle iniziali.

4. Esperimenti e discussioni

In questo paragrafo verranno discussi e commentati i risultati ottenuti. L'algoritmo di cui sopra, insieme ad alcune versioni ottimizzate dell'algoritmo di De Bonet sono stati implementati in MatLab 5 su un Pentium II 350 Mhz, 64Mb. Ovviamente tutte le prove sono state eseguite sulla stessa piattaforma. In particolare ci si è giovati di tessiture di *input* aventi dimensione 64x64. La nostra fase di analisi dei *patterns* richiede in media più tempo della corrispondente fase di analisi dell'algoritmo di De Bonet ma il nostro approccio va molto più veloce nella fase di sintesi. Inoltre tessiture di diverse dimensioni possono essere facilmente generate non appena il corrispettivo *frequency tree* è stato calcolato e registrato. La Tabella 1 mostra i tempi medi di esecuzione ottenuti nella nostra implementazione dei due algoritmi.

	Our Approach	De Bonet	Mixed
Analysis	*26″*	*4″*	*26″*
Synthesis of 64x64	*3″*	*140″*	*30″*
Synthesis of 128x128	*15″*	*550″*	*122″*

Tabella 1

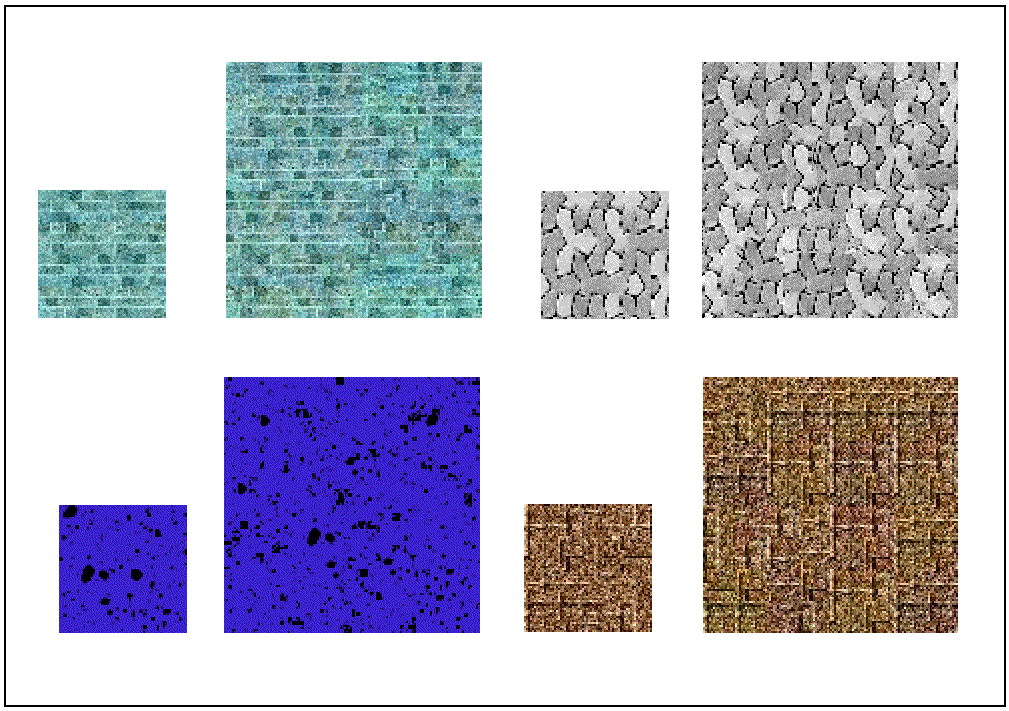

Fig. 4.

Una tipica tessitura ricostruita con il nostro approccio a partire dall'*input* in Fig.3.a o in Fig.3.e è confrontata con la sintesi ottenuta con l'algoritmo di De Bonet in Fig.3.b/3.c e in Fig.3.f/3.g. Risulta evidente come il risultato ottenuto con il nostro metodo sia buono, anche se una leggera perdita di qualità può essere osservata per esempio confrontando Fig.3.b rispetto alla Fig.3.c. Abbiamo osservato che un'ottima qualità della tessitura sintetizzata si ottiene utilizzando un approccio misto: invece di campionare *random*, da un *cluster* per ottenere i valori Laplaciani RGB, cerchiamo di operare una scelta più mirata alla qualità visivo-percettiva del *pixel* in quel contesto. Si cerca cioè di individuare ina maniera effettiva la *migliore* scelta di valori RGB, nel senso considerato da De Bonet per i *parent vectors.* I tempi di tale implementazioni sono riportati nell'ultima colonna della Tabella 1. Esempi di sintesi di tessiture ricostruite con questo approccio

sono riportati in Fig.3.d e in Fig. 3.h. Risulta più che evidente che questi ultimi risultati sono della stessa qualità di quelli ottenuti in Fig.3.c e in Fig.3.g. Altre tessiture ottenute con tale approccio "misto" sono riportate nella Fig.4.

Uno dei fattori cruciali che determina l'espressività intrinseca del metodo multirisoluzione proposto sembra essere legato ai filtri applicati prima della clusterizzazione. Una scelta non oculata e precisa porterebbe infatti solo a risultati parzialmente soddisfacenti. D'altronde questa considerazione non è del tutto nuova: Zhu e Mumford (1997) hanno sempre palesato la necessità di includere nel processo di apprendimento, una sorta di discriminazione dei filtri da applicare basati su considerazioni legati all'entropia o comunque alla teoria dell'informazione. Meno cruciale invece appaiono, sia l'algoritmo scelto per il *clustering* sia il numero di *cluster* scelto per ciascun livello. Tutti gli esperimenti riportati sono stati realizzati, partizionando l'insieme dei *pixel* di ciascun livello, in 4/5 *cluster*. Un più alto numero di *cluster* non ha dato ulteriori migliorie, come d'altronde con un numero inferiore di *cluster* si sono avuti risultati del tutto insoddisfacenti.

5. Conclusioni

In questo capitolo si è presentato un efficiente algoritmo multirisoluzione per la sintesi automatica di immagini reali di tipo *texture*. I risultati sperimentali discussi e mostrati ne dimostrano l'effettiva bontà e utilità della metodologia adottata. Sviluppi futuri di tale ricerca prevedono la possibilità di utilizzare la stessa idea nella discriminazione di tessiture, riguardanti per esempio immagini di tipo SAR.

In particolare è nostra intenzione, riuscire a confrontare sperimentalmente nell'ambito della classificazione/segmentazione di tessiture il nostro metodo con i paradigmi *standard* presenti in letteratura.

Capitolo 4

Metodi Entropici per la Costruzione di Mappe di Salienza

In questo Capitolo vengono descritte delle tecniche volte ad apprendere le caratteristiche più rilevanti, da un punto di vista strettamente legato alla teoria dell'informazione, dei *pixel* di una data immagine, con lo scopo di ottenerne un cosiddetto valore di salienza. Tale metodologia di lavoro tenta di combinare insieme metodi di elaborazioni delle immagini basati sull'*entropia* di Shannon e tecniche per lo "smussamento" statistico di dati irregolari e/o sparsi (Yager - 1991) ottenendo degli algoritmi per certi versi analoghi a quelli basati sull'Analisi delle Componenti Principali, in breve PCA (Joliffe - 1986). Esempi ed applicazioni sono inoltre presentati e discussi.

1. Introduzione

Il problema della costruzione di Mappe di Salienza da un'immagine digitale è stata oggetto di una attenzione via via crescente da parte della comunità scientifica. La salienza *S(x,y)*, relativa ad un dato oggetto, è una misura della probabilità che il dato *pixel (x,y)*, con intensità *I(x,y)*, segua in qualche modo la traccia dell'oggetto sotto osservazione. Ciò significa che a più alti valori di salienza corrisponde una più alta probabilità che il dato *pixel* appartenga a quell'oggetto. Si parla in questo caso di *Visual Attention*: l'azione di operatori visuali di alto livello si restringe ad un sottoinsieme del campo visuale detto fuoco dell'attenzione (FOA - *Focus Of Attention*). Le Mappe di Salienza sono utili *tool* per la preliminare individuazione di sottoinsiemi dell'immagine originale su cui concentrare l'attenzione di più sofisticati e raffinati algoritmi di *detection* che risultano spesso essere decisamente più pesanti dal punto di vista computazionale. Evidenze sperimentali nel campo delle scienze cognitive (Palmer - 1983; Moghaddam - 1997) hanno mostrato come tale approccio in due fasi, determinazione di una zona di attenzione ed elaborazione specifica di più alto livello, simuli proprio il comportamento effettivo del sistema percettivo umano.

In generale esistono due approcci fondamentali per la costruzione di Mappe di Salienza: basati sulla scena o basati sull'immagine (*scene-based* e *image-based*): nel primo approccio si fa uso di informazioni di carattere generale sull'immagine note a priori che vengono integrate nell'analisi. D'altra parte il secondo approccio tenta di classificare i *pixel* dell'immagine in accordo con proprietà statistiche che possono essere direttamente misurate ed individuate nell'immagine stessa.

L'approccio qui presentato apprende la più rilevante combinazione di caratteristiche misurate direttamente su ogni *pixel* e ne determina un valore di salienza.

Un recente metodo basato sulle immagini detto approccio *eigenfaces* (Turk - 1991) realizza ciò nella seguente maniera: un insieme esaustivo di caratteristiche o *feature* relative ad ogni *pixel* viene misurato (generalmente grazie ad una finestra scorrevole di una fissata dimensione centrata sul *pixel* stesso) su una data classe di esempi di apprendimento. Successivamente con una trasformata di Karuhnen-Loeve, eseguita sull'insieme di dati delle caratteristiche apprese, si riescono ad identificare le componenti principali (corrispondenti ai più alti autovalori della matrice di covarianza) che possono essere quindi usate per il riconoscimento.

Tale tecnica di base è stata successivamente oggetto di un ulteriore raffinamento (Moghaddam - 1997) trovando applicazione specifica nella determinazione di Mappe di Salienza in presenza di distribuzioni Gaussiane, Gaussiane miste e rumore di vario tipo.

Il nostro approccio è simile ma più efficiente dal punto di vista strettamente computazionale; la funzione di salienza è appresa da un insieme di esempi positivi o di *training* ed è ottenuta come una combinazione lineare pesata di tutte le caratteristiche misurate per ogni *pixel*. I pesi sono ottenuti in funzione dell'*entropia* della distribuzione regolarizzata di ciascuna delle caratteristiche misurate sui dati di *training* e sono inoltre strettamente legati alla distribuzione spaziale attorno ad ogni *pixel*. Il metodo proposto risulta essere efficiente e gli esperimenti preliminari fin qui condotti mostrano una buona *performance*

complessiva nonché l'effettivo apprendimento di valori di salienza in diversi ambiti applicativi.

2. Costruzione di Mappe di Salienza

In questa Sezione descriviamo le idee di base di un algoritmo da noi proposto per la costruzione di Mappe di Salienza. Esso è articolato in due fasi e precisamente una prima fase di *learning* cioè l'apprendimento da un insieme di esempi positivi e una seconda fase in cui si procede alla costruzione vera e propria della Mappa di Salienza.

Assumiamo di avere un immagine digitale ***I***, rappresentata come una matrice $m_x n$ contenente le informazioni relative all'intensità $\boldsymbol{I}(x,y)$ $x=1,2,\ldots,m$ $y=1,2,\ldots,n$. Per semplicità assumiamo che $\boldsymbol{I}(x,y)$ sia una funzione scalare (a livelli di grigio) ma i metodi presentati possono facilmente essere generalizzati ed adattati al caso di funzioni a più valori come immagini a colori o rappresentazioni multiscala, ecc..

La scelta del più idoneo insieme di *feature*, come noto, è uno dei fattori chiave di un qualsiasi algoritmo di riconoscimento. La nostra tecnica consente di valutare insiemi di *feature* relativamente grandi con un moderato carico sulle risorse computazionali richieste. In ogni caso, tale insieme risulta comunque essere un parametro cruciale dell'algoritmo e deve essere fornito in *input*. Ulteriori estensioni e studi in questo senso verranno descritti in dettaglio in seguito.

E' comunque preferibile considerare un grosso numero di *feature*, senza avere la necessità di prendere preliminari decisioni sulla rilevanza effettiva delle singole

caratteristiche misurate in quanto alla fine l'algoritmo assegnerà automaticamente il giusto peso a ciascuna di esse.

L'**input** dell'algoritmo è quindi costituito da una serie di *K* esempi positivi E_1, E_2, …, E_k dell'oggetto da riconoscere, da un insieme di *feature* F_1, …, F_h, insieme con un'immagine da analizzare ***I***.

L'**output** è una Mappa di Salienza *S(x,y)* su ***I*** .

2.1 Fase di *Learning*

La fase di apprendimento da un insieme di esempi positivi ricava tutte le informazioni di base necessarie per costruire una funzione di salienza come combinazione lineare delle diverse *feature* considerate, in accordo con il criterio della minima *entropia*. Gli esempi positivi sono sottoinsiemi di *pixel* appartenenti all'oggetto da riconoscere. Essi potrebbero essere ottenuti in maniera interattiva chiedendo all'utente di selezionarli da immagini digitali appartenenti alla stessa classe dell'immagine da analizzare.

Il processo di *learning* si realizza misurando F_1,…, F_h per ogni *pixel* di ciascun esempio positivo. Tali informazioni vengono poi organizzate in un insieme di istogrammi, ciascuno relativo a una data *feature*. Indichiamo con $Histo(F_j)(v)$ la frequenza osservata del valore *v*, della *feature* F_j, sull'universo degli esempi positivi dei *pixel*.

In particolare, negli esperimenti da noi realizzati, abbiamo scelto per ogni oggetto da riconoscere, una maschera rettangolare, di opportune dimensioni, centrata su un *pixel*. Come *feature*, almeno in questa fase preliminare si sono considerate i livelli di intensità in ogni punto della maschera. Questa scelta

ovviamente impone delle limitazioni sia sulla grandezza che sul tipo vero e proprio di oggetti che il metodo è capace di riconoscere: devono essere di piccola dimensione e la distribuzione spaziale dei *pixel* attorno all'oggetto non deve essere soggetta a brusche variazioni di intensità. Differenti scelte e strategie sono comunque possibili e sono tuttora soggette a specifiche ricerche.

L'istogramma $Histo(F_j)(v)$ relativo alla proprietà F_j fornisce rilevanti informazioni di carattere statistico circa la distribuzione F_j sui *pixel* dell'oggetto. Effetti *random*, rumore e altri inevitabili disturbi, cui va aggiunto il fatto che si ha a che fare comunque con un numero limitato di dati disponibili, introducono degli elementi di disturbo che agiscono in maniera imprevedibile nel processo di formazione degli istogrammi. Si è quindi pensato di alleviare questo *side-effect*, smussando gli istogrammi attraverso una convoluzione con una funzione Gaussiana di dimensione opportuna, con il dichiarato obiettivo di rendere più evidenti i picchi effettivi riducendone le asperità. Tale obiettivo può comunque essere raggiunto utilizzando altre tecniche note in letteratura. Per esempio, tecniche come quelle discusse in (Yager - 1991) danno *performance* migliori ma non sono state qui considerate per semplicità.

Denotiamo l'istogramma regolarizzato relativo alla feature F_j con il simbolo $HistoR(F_j)$.

L'algoritmo in questione pesa opportunamente ciascuna *feature* F_j tenendo esplicitamente conto di due importanti fattori. In primo luogo si sfruttano le conoscenze note *a priori* circa lo specifico problema di riconoscimento. L'utente

può conoscere *a priori* il grado di rilevanza specifica di ciascuna *feature* F_j. Per esempio il livello di grigio in un punto della maschera "lontano" dal centro è meno rilevante del livello di grigio del centro medesimo, così come l'intensità relativa al colore blu lo sarà meno rispetto alla luminosità, ecc..

Il secondo fattore è più legato alla capacità effettiva di ciascuna *feature* F_j di rilevare i *pixel* dell'oggetto. L'idea che le caratteristiche rilevanti mostrano un più alto grado di correlazione suggerisce in maniera del tutto naturale di utilizzare l'*entropia* di Shannon come indicatore di base. Noi quindi proponiamo di calcolare, dai cosiddetti istogrammi regolarizzati, l'*entropia* H_c della distribuzione di ogni caratteristica F_j.

L'assunzione che permette di giustificare questo approccio sta nel fatto che *feature* rilevanti per l'oggetto sotto osservazione, se misurate sui *pixel* dell'oggetto assumeranno una qualche regolarità statistica, saranno cioè poco soggette a variazioni *random* o a rumore, mostrando in maniera netta una marcata preferenza per un non troppo largo *range* di valori specifici. Queste ipotesi non sono sempre vere e realistiche, ma noi pensiamo che quando ciò si realizza, l'*entropia* di Shannon ne è un ottimo indicatore. In particolare, se l'entropia assume valori alti ciò sta a significare che la data *feature* è affetta da un'incertezza diffusa, dovuta possibilmente a cause più o meno *random*; ciò significa che non si può utilizzare tale *feature* come "segno" reale dell'oggetto in questione o quantomeno bisogna assegnare ad essa una rilevanza minore nella determinazione di un valore di salienza.

Proponiamo allora per ogni *feature* F_j un fattore di peso w_j, la cui espressione funzionale è la seguente:

$$w_j = f(j) * (K H_j)^{-1} \qquad (1)$$

dove la *f(j)* incorpora le informazioni sulla cosiddetta *a priori knowledge* circa il problema in oggetto, ed è quindi una funzione definita dall'utente, *K* è un'opportuna costante positiva ed H_j indica l'*entropia* di Shannon della variabile casuale la cui distribuzione statistica è data dall'istogramma regolarizzato $HistoR(F_j)$.

L'espressione funzionale di H_j è la seguente (Ash - 1965; Shannon 1948):

$$H_j = \sum_t - HistoR(F_j)(t) Log\,(HistoR(F_j)(t)) \qquad j=1,2,\ldots,h \qquad (2)$$

dove l'indice *t* si riferisce all'insieme dei valori osservati per la data feature F_j.

Si osservi che se come *feature* scegliamo i livelli di grigio misurati, nella maschera di *detect*, così come discusso sopra, si può assumere come *f(j)* una funzione inversamente proporzionale alla distanza del punto considerato dal centro della maschera. E' questa la scelta seguita nei nostri esperimenti.

La determinazione dei pesi w_j conclude la fase di *learning*. Come commento finale alla fase di apprendimento possiamo affermare che il metodo proposto è più economico in termini di tempo e di spazio rispetto al più classico approccio del PCA e permette quindi l'elaborazione effettiva di un grosso numero di esempi positivi. I risultati preliminari come vedremo più in dettaglio nelle successive sezioni risultano essere incoraggianti.

2.2 Calcolo Effettivo della Mappa di Salienza

La seconda fase dell'algoritmo semplicemente calcola *S(x,y)* per ogni pixel dell'immagine di input *I*, secondo la seguente espressione funzionale:

$$S(x,y) = \sum_j w_j \, G(F_j, HistoR(F_j), x, y) \qquad j=1, 2, \ldots, h \tag{4}$$

dove:

$$G\left(F_j, HistoR\ \left(F_j\right), x, y\right) = \left(1 + \frac{\left|F_j(x,y) - HistoR\ \left(F_j\right)^{MAX}\right|}{Z_j}\right)^{-1} \tag{5}$$

e Z_j è un opportuna costante legata al valore modale della *feature* F_j. Si osservi come il processo di apprendimento (Fase 1) non debba essere ripetuto in presenza di nuove immagini da processare.

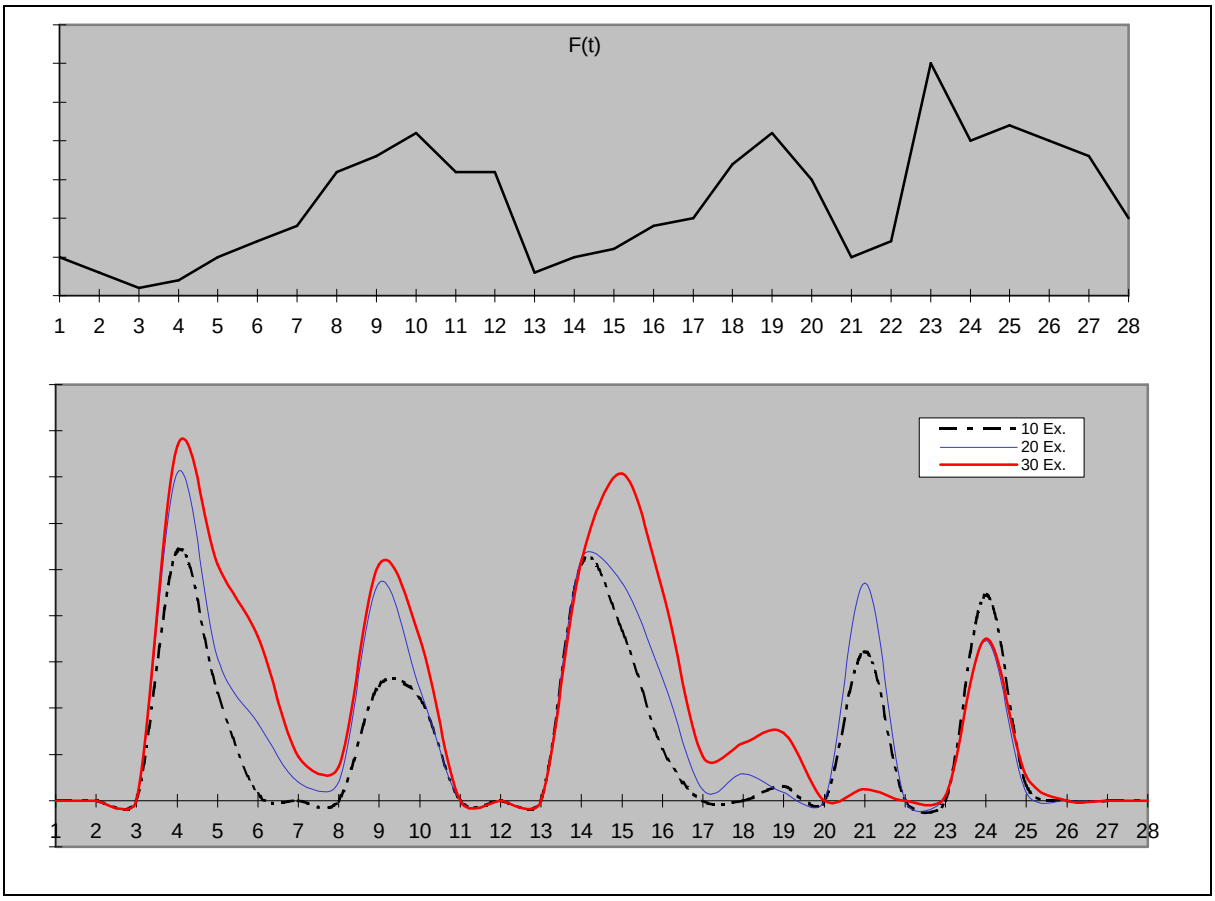

Fig. 1 - La funzione F(t) e come in legenda, le tre successive mappe di salienza ottenute nella ricerca dei punti stazionari.

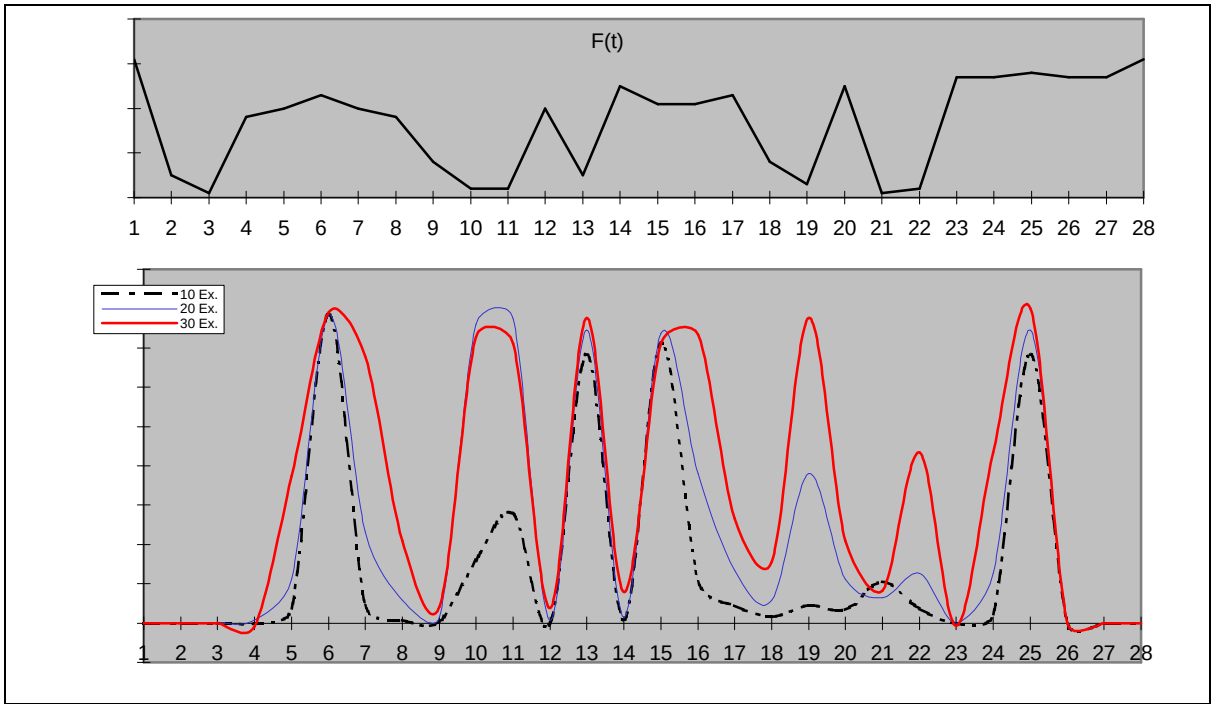

Fig. 2 - La funzione F(t) e come in legenda, le tre successive mappe di salienza ottenute nella ricerca dei punti in cui la funzione è crescente.

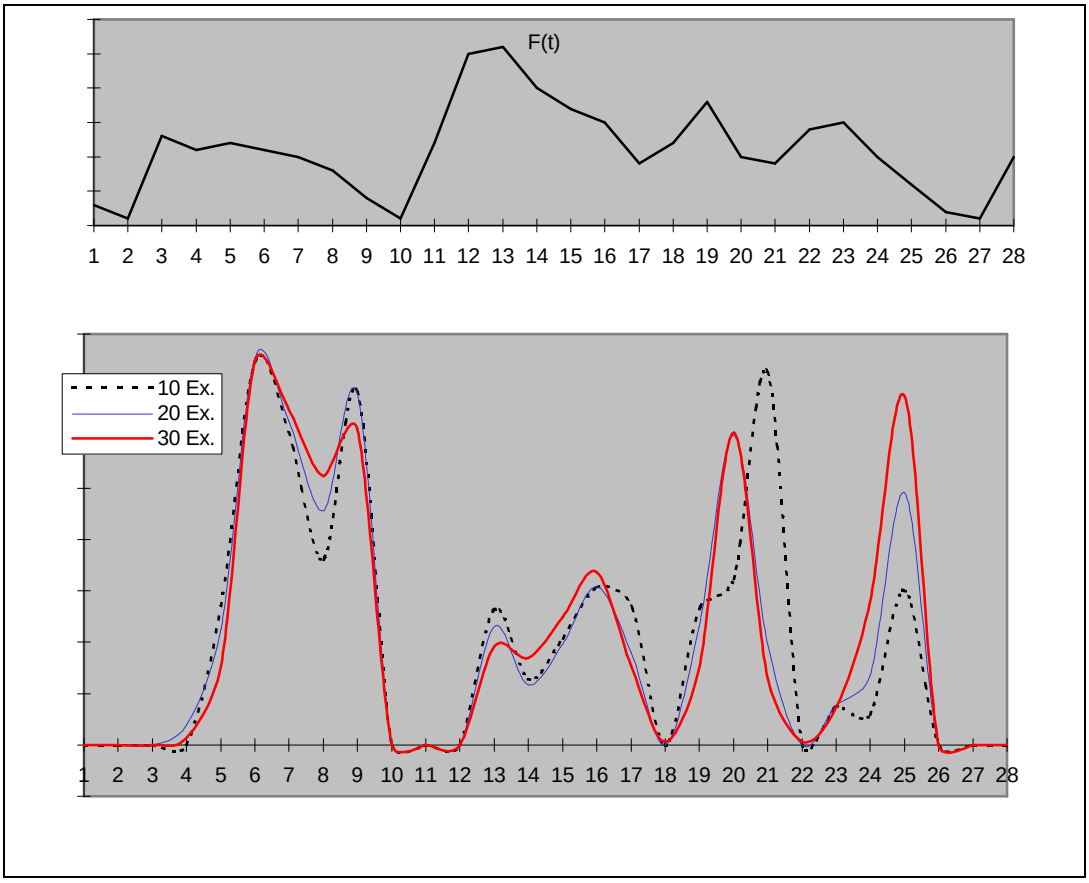

Fig. 3 - La funzione F(t) e come in legenda, le tre successive mappe di salienza ottenute nella ricerca dei punti in cui la funzione è decrescente.

3. Risultati sperimentali

Anche se la tecnica proposta è ovviamente rivolta al mondo 2D delle immagini digitali, i nostri primi esperimenti sono stati compiuti su segnali unidimensionali al fine di meglio comprendere ed analizzare i dettagli, i punti di forza e le "debolezze" che comunque risultano già evidenti.

Il nostro primo *test* è stato eseguito su un segnale discreto unidimensionale *F(t)*. In questo caso gli oggetti che abbiamo voluto riconoscere su di esso in esperimenti successivi, sono stati i punti stazionari (estremi o flessi), punti di crescenza e punti di decrescenza. Ovviamente queste informazioni possono essere

analizzate e scovate attraverso lo studio della derivata prima e lo scopo di questo semplice esperimento è stato proprio quello di mostrare come ciò possa essere appreso facilmente utilizzando l'algoritmo di *learning* appena descritto. Abbiamo eseguito le prove utilizzando *training set* di esempi positivi rispettivamente con 10, 20 e 30 esempi positivi; le relative mappe di salienza sono riportate in Fig.1, 2 e 3.

Le *feature* utilizzate per il processo di ricerca o di *recognition* sono state in questo caso il valore della funzione e le differenze prime in un piccolo intorno del *pixel* in questione. L'idea che sta dietro a questa particolare scelta, riguarda appunto il fatto che l'oggetto di interesse è interamente caratterizzabile sfruttando le informazioni contenute nelle differenze che diventano quindi dei significativi parametri del sistema. E' possibile osservare la buona corrispondenza della funzione di salienza con l'oggetto di interesse. A meno di non considerare come feature anche caratteristiche più articolate è impossibile riuscire a determinare oggetti più complessi come punti di concavità, convessità, cuspidi, ecc.

L'ulteriore estensione al caso 2D, cioè alle prime immagini è presentata negli esempi successivi. La Fig. 4 infatti mostra i risultati e cioè la relativa mappa di salienza ottenuta per la ricerca di linee verticali di un dato tono di grigio. In particolare in Figura 4, la prima colonna mostra l'immagine di input, la seconda e la terza mostrano la relativa Mappa di Salienza ottenute rispettivamente con 100 e con 200 esempi positivi. In questo caso la mappa di salienza, viene rappresentata su uno spazio bidimensionale, come un'immagine; ciò significa che sono i valori più chiari ad indicare un maggior grado di probabilità di appartenenza all'oggetto che si sta analizzando. Ovviamente si tratta di un caso relativamente semplice. In

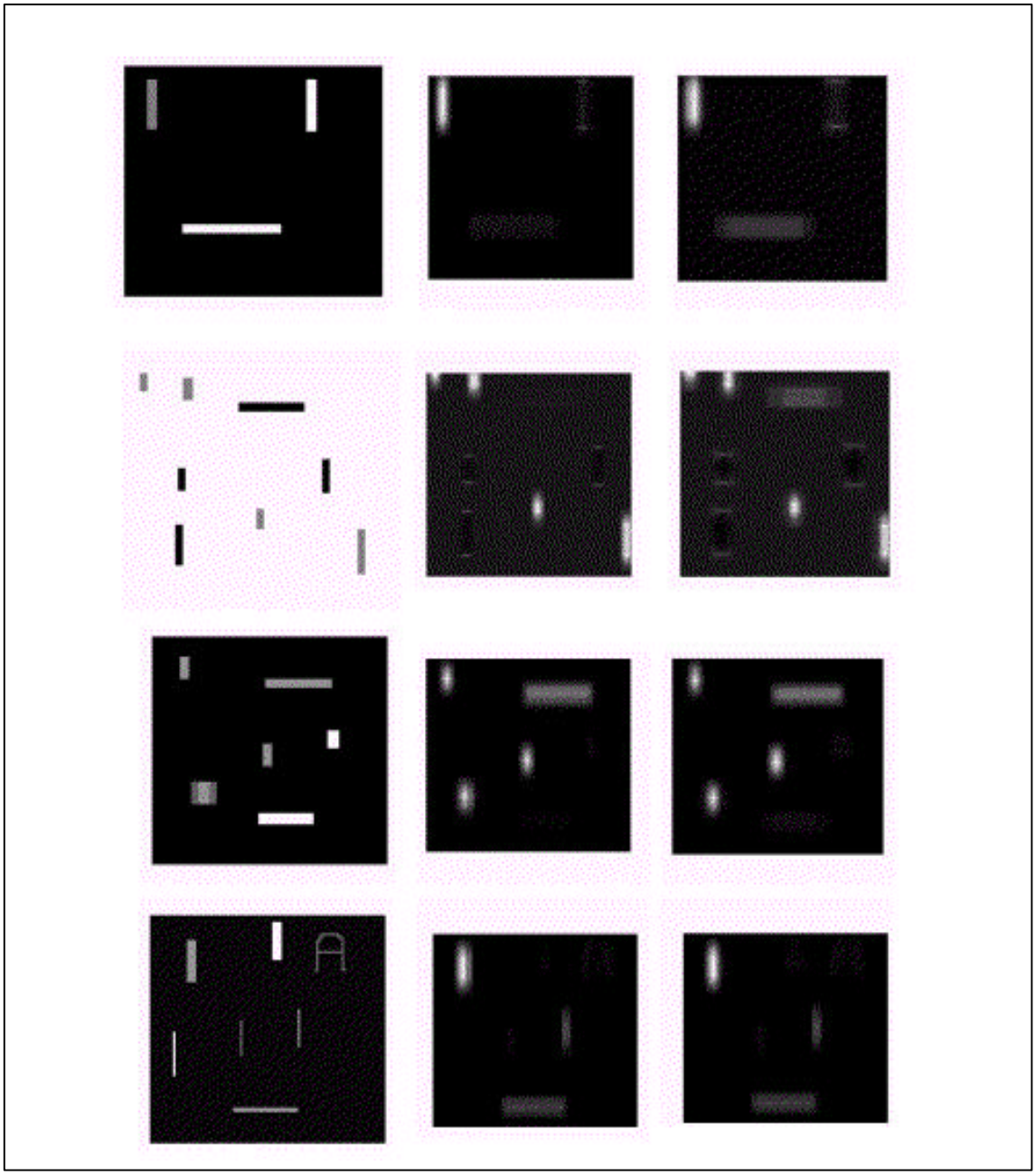

Fig.4. - Algoritmo per la costruzione di Mappe di Salienza applicato alla ricerca di linee verticali di un dato tono di grigio

questo momento si sta però lavorando per un'ulteriore estensione di quest'approccio a casi più generali.