UNIVERSITA' DEGLI STUDI DI CATANIA

FACOLTA' DI INGEGNERIA

Dipartimento di Ingegneria Informatica e delle Telecomunicazioni

IGNAZIO COCO

TECNICHE DI CLUSTERING E COLOR PROCESSING PER FIRE DETECTION

TESI DI LAUREA

Relatore:
Chiar.mo Prof. Ing. Alberto Faro

Correlatore:
Ing. Concetto Spampinato

Anno Accademico 2005 - 2006

SOMMARIO

Capitolo 1 – Introduzione

Capitolo 2 – Sistemi di Fire Detection

2.1 Rilevamento incendio intelligente in real-time basato su elaborazione video

2.2 Rilevamento fuoco basato sulla visione

2.3 Rilevamento degli incendi nei tunnel usando l'elaborazione di immagini

2.4 Rilevamento incendio in real-time

Capitolo 3 – Algoritmi di Clustering

3.1 Generalità sugli algoritmi di clustering

3.1.1 Introduzione

3.1.2 Obiettivo degli algoritmi di clustering

3.1.3 Classificazione degli algoritmi di clustering

3.1.4 Misura della distanza - Minkowski Metric

3.1.5 Applicazioni

3.2 Conceptual Clustering

3.3 Hierarchical Clustering

3.3.1 Generalità dell'algoritmo

3.3.2 Single-Linkage Clustering: l'algoritmo

3.3.3 Problemi

3.4 Fuzzy C-Means Clustering

3.5 Mixture of Gaussians

3.6.1 Introduzione agli algoritmi di clustering basati su modello

3.6.2 L'algoritmo Mixture of Gaussians

3.6 K-means Clustering

Capitolo 4 – Color Space

4.1 Lo spazio RGB

4.2 Lo spazio HSV

4.3 Lo spazio HSB

4.4 Il cromaticity space

Capitolo 5 – Sistema Implementato

5.1 Introduzione

5.2 Creazione del modello

3.2.1 Funzionamento

3.2.2 Selezione dei campioni

3.2.3 Trasformazione delle immagini

3.2.4 Creazione della matrice modello

5.3 Fase di test: Fire Detection

5.3.1 Ottimizzazione dell'immagine

5.3.2 Filtraggio dei pixel

5.3.3 Clusterizzazione dell'immagine

5.3.4 Elaborazione dei cluster

5.4 Stima dei risultati e notifica degli allarmi

5.4.1 Risultati

5.4.2 Allarmi locali

5.4.3 Allarmi remoti

5.5 Risultati sperimentali

5.5.1 Immagini contenenti fuoco

5.5.2 Immagini non contenenti fuoco

Capitolo 6 – Implementazione Software

6.1 Modello

6.1.1 Limitazione Punti Modello (Limita.m)

6.1.2 Trasformazione RGB – Cromaticity Space (S_T_trasf.m)

6.1.3 Creazione Modello (Creagrafico.m)

6.2 Analisi Immagini

6.2.1 Analisi sequenza immagini (Ciclo.m)

6.2.2 Analisi di un'immagine (Analisi.m)

6.2.3 Ottimizzazione immagine (Ottimizza.m)

6.2.4 Filtraggio immagine (Filtra_Pixel.m)

6.2.5 Clustering immagine (Trova_Cluster.m)

6.2.6 Analisi cluster (Analizza_Cluster.m)

6.2.7 Analisi dei pixel (Test_Image.m)

6.2.8 Invio di allarmi (Invia_Allarme.m)

Capitolo 7 – Conclusioni e sviluppi futuri

Bibliografia

Ringraziamenti

CAPITOLO I

Introduzione

Il patrimonio forestale italiano, tra i più importanti d'Europa per ampiezza e varietà di specie, costituisce un'immensa ricchezza per l'ambiente e l'economia, per l'equilibrio del territorio, per la conservazione della biodiversità e del paesaggio. Tuttavia ogni anno assistiamo all'incendio di migliaia di ettari di bosco, molto spesso dovuto a cause dolose, legate alla speculazione edilizia, o all'incuria e alla disattenzione dell'uomo. Le conseguenze per l'equilibrio naturale sono gravissime e i tempi per il riassetto dell'ecosistema molto lunghi.

Nonostante l'istituzione di varie campagne di sensibilizzazione e grazie a una migliore organizzazione del complesso apparato antincendio delle Regioni e dello Stato, il rischio rimane comunque molto alto: ogni anno circa 77.000 ettari di terreno vengono bruciati.

Il problema degli incendi nei boschi può essere affrontato nel momento in cui questo si verifica, un fattore molto importante è quindi la tempestività con cui si interviene per cercare di contenere tale fenomeno. Durante la stagione estiva nonostante l'immane lavoro svolto da Vigili del Fuoco, Corpo Forestale e Protezione Civile, i danni rilevati dopo che un incendio viene spento sono comunque enormi.

Per essere in grado di limitare tali danni, occorrerebbe un sistema di sensori posizionati in punti strategici per poter intervenire con celerità ed evitare

l'evolversi del fenomeno stesso. Purtroppo gli unici sensori di questo tipo sono quelli usati per gli edifici che permettono di rilevare temperature alte e la presenza di fumi sviluppati dal processo di combustione. Questi sensori, seppur molto efficienti in ambiente indoor, non possono essere utilizzati in ambiente esterno perché la loro sensibilità non è tale da poter rilevare un incendio ad una grande distanza, inoltre per coprire un'area molto vasta occorrerebbe un numero di sensori molto elevato che renderebbe il sistema molto costoso.

Un'alternativa che soddisfa i vincoli sopra esposti, è la "Fire Detection" tramite tecniche di video processing. Con questa tecnica è possibile rilevare un incendio semplicemente analizzando delle immagini provenienti da una fonte di acquisizione video. Questo metodo già usato con successo in svariati ambienti permette di determinare la presenza di un incendio anche su aree molto vaste sfruttando un dispositivo di acquisizione video (una webcam, una videocamera di sorveglianza, etc.). Le camere usate per l'acquisizione hanno il vantaggio di poter essere posizionate in zone strategiche (una posizione sopraelevata), per monitorare aree molto vaste sfruttando appunto un unico dispositivo. Inoltre tale tecnica risulta molto efficiente perché indipendente da condizioni climatiche e dalla luminosità stessa della zona sorvegliata. Il sistema sviluppato si basa su questa tecnica, sfruttando tecniche di clustering

per migliorare la probabilità di rilevamento incendi, ed il cromaticity space per rendere l'analisi della scena invariante alla luminosità e, contemporaneamente, per ridurre la complessità computazionale del sistema stesso.

CAPITOLO II

Sistemi di Fire Detection

2.1 Rilevamento incendio intelligente in real-time basato su elaborazione video

In [1] Thou-Ho, Chen, Ping-Hsueh Wu e Yung-Chuen Chiou hanno sviluppato un sistema che si basa su un processo in 2 stadi. Nel primo stadio si cerca un eventuale incendio in un'immagine estrapolando da essa i pixel di fuoco. Tali pixel vengono riconosciuti per confronto con un modello RGB, che è scelto per far si che il tempo di computazione sia basso.

Il primo stadio viene semplicemente realizzato introducendo una serie di regole per permettere una distinzione tra i pixel di fuoco ed i pixel del background

- regola 1: $R > R_T$
- regola 2: $R >= G > B$
- regola 3: $S >= ((255-R)*S_T/R_T)$

I pixel che all'uscita del primo stadio verranno riconosciuti come fuoco, saranno tutti quelli che soddisfano i tre requisiti sopraindicati: alcuni di questi però potranno essere pixel che pur rispettando queste regole non sono effettivamente di fuoco. Ad esempio, aree attorno all'incendio potrebbero assumere colori simili al fuoco, oppure alcune aree dello sfondo potrebbero avere componenti cromatiche che soddisfano i vincoli precedenti.

Per ovviare a questo problema nel secondo stadio i pixel vengono filtrati valutando le caratteristiche dinamiche del fuoco.

Nel secondo stadio viene valutato il fattore di crescita dei pixel di fuoco, se tale valore supera una certa soglia prestabilita verrà generato un allarme. Infatti gli eventuali pixel che vengono riconosciuti come 'fuoco' ma che in realtà non lo sono, verranno scartati poiché non rispettano le caratteristiche dinamiche dell'incendio. Un'area colpita da un incendio presenterà dei movimenti 'istantanei' e delle oscillazioni che le aree statiche non manifestano. Tale proprietà viene estrapolata andando a valutare la differenza tra due o più frame consecutivi presi da una sorgente video. Per limitare i falsi allarmi il secondo stadio viene applicato diverse volte durante la durata del fenomeno.

2.2 Rilevamento fuoco basato sulla visione

Questo metodo [2] implementato da Che-Bin Liu e Narendra Ahuja prevede l'uso di tre differenti modelli applicati ad una sorgente video per riconoscere un incendio. Sostanzialmente il processo si suddivide in tre fasi, in ognuna di queste fasi vengono effettuate opportune trasformazioni dell'immagine e confronti col modello relativo. Nel primo passo si cercano le potenziali

regioni di fuoco usando modelli spaziali e spettrali del fuoco. Vengono individuate le regioni ad alta luminosità nell'immagine in scala di grigi e queste regioni vengono espanse in direzione dei pixel che hanno colori tipici del fuoco. Si verifica poi che ogni regione ad alta luminosità sia racchiusa da una zona con colori ignei. In questo modo si prevengono quelle zone che includono solo sorgenti luminose. Nel secondo passo vengono individuati i contorni di queste regioni tramite i coefficienti di Fourier ed infine nel terzo passo viene valutato il modello AR dell'immagine attuale con il modello AR dell'immagine precedente per mettere in evidenza l'evoluzione dinamica della regione. Viene valutata una funzione che permette di distinguere le componenti a bassa frequenza tipiche del fuoco, che sono costanti, da quelle ad alta frequenza che sono variabili. Infine i parametri d'uscita del modello AR e dei coefficienti di Fourier vengono usati come valori di ingresso al classificatore delle regioni

2.3 Rilevamento degli incendi nei tunnel usando l'elaborazione di immagini

Nel lavoro di S. Noda [3] illustra il funzionamento di due sistemi di rilevamento incendio basati su tecniche di elaborazione di immagini sfruttando sistemi di videosorveglianza preesistenti.

La prima tecnica consente l'uso di telecamere a colori. Usando il sensore CCD di una telecamera a colori è possibile trovare una relazione che lega il rapporto G/R (green / red) con la temperatura di una porzione dell'immagine in esame. In questo modo è possibile determinare quando la zona monitorata è colpita da un incendio, confrontandola con i dati rilevati in condizioni normali.

La seconda tecnica si basa sull'uso di sistemi di videocamere di sorveglianza in bianco e nero. Questo metodo si basa fondamentalmente sul confronto dell'istogramma dell'immagine in scala di grigi con l'istogramma 'modello'. Quest'ultimo non è altro che un istogramma ricavato in condizioni di traffico normali. Per aumentare l'efficienza di questo sistema, l'istogramma viene anche confrontato con l'istogramma di immagini d'archivio in cui è presente un incendio.

2.4 Rilevamento incendio in real-time

In questo sistema [4], Healey presenta una tecnica di rilevamento incendi molto efficiente dal punto di vista computazionale. Infatti ogni incendio può essere monitorato nella sua evoluzione in tempo reale. Ogni frame della sorgente video viene elaborato in quattro stadi. Nel primo stadio, (sistema offline) viene creata una griglia della scena inquadrata in base alla geometria della zona ed alla qualità del sistema di acquisizione video. Successivamente, ogni elemento della griglia viene processato e schedulato come 'fuoco' o 'non fuoco'. Questo passo, viene applicato valutando le caratteristiche fisiche di ogni pixel e confrontandole con un modello realizzato usando un notevole numero di dati video. Nel passo successivo tutti gli elementi della griglia adiacenti riconosciuti come fuoco vengono accorpati in un'unica regione. Infine nell'ultimo passo vengono confrontate le zone di fuoco del frame attuale con quelle del frame precedente potendo così stabilire la presenza di un incendio, il fattore di crescita e la sua evoluzione temporale. Il sistema può quindi decidere se entrare in uno stato di allarme (monitorizzazione dell'evento), stato di soppressione (evento riconosciuto ed attivazione delle contromisure) o continuare l'operazione di controllo.

CAPITOLO III

Algoritmi di clustering

3.1 Generalità sugli algoritmi di clustering

3.1.1 Introduzione

Il clustering può essere considerato il principale problema di riconoscimento non supervisionato, nello stesso modo in cui potrebbe esserlo qualsiasi altro problema in cui l'obiettivo è di trovare una certa struttura in una collezione di dati non classificati.

Una semplice definizione di clustering potrebbe essere: "il processo di organizzazione di un set di oggetti in un numero arbitrario di gruppi, i cui membri sono simili per qualche aspetto". Un cluster è quindi una collezione di oggetti che sono "simili" tra loro (secondo certe caratteristiche) e che sono "diversi" da oggetti appartenenti agli altri cluster.

Il processo di clustering è anche chiamato "analisi della segmentazione" o "analisi della tassonomia", è un modo per creare gruppi di oggetti, o appunto cluster, in modo che i profili degli oggetti appartenenti ad uno stesso cluster siano simili tra loro, ed invece, i profili degli oggetti tra cluster diversi siano abbastanza distinti.

L'analisi dei cluster può essere effettuata su diversi tipi di oggetti. Ad esempio, l'insieme dei dati da classificare potrebbe essere costituito da un certo numero di osservazioni su alcuni soggetti di un determinato studio, in

cui ogni osservazione contiene un insieme di variabili del fenomeno osservato.

Qui di seguito viene presentato un semplice esempio di come agisce un algoritmo di clustering: i dati dell'esempio sono rappresentati dai punti (caratterizzati dalle loro coordinate cartesiane), l'algoritmo usa la distanza euclidea come criterio di similarità tra i dati.

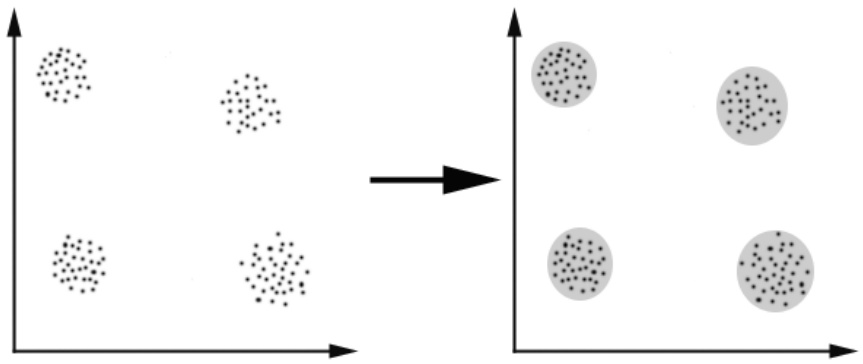


Figura 1 - A sinistra i dati del problema in esame, a destra vengono evidenziati i gruppi

Nel caso in esame, possiamo facilmente notare che l'insieme dei dati può essere intuitivamente diviso in 4 gruppi; come detto, il criterio di similarità in questo caso è la distanza: due o più oggetti appartengono allo stesso cluster se sono "vicini" o "simili" secondo una data distanza (nel nostro caso la distanza considerata è quella geometrica). Questo tipo di clustering viene chiamato "distance-based clustering".

Un altro tipo di clustering, è il clustering concettuale: 2 o più oggetti appartengono ad uno stesso gruppo, se uno di essi contiene una nozione comune agli altri oggetti del cluster. In altre parole, gli oggetti vengono raggruppati in modo che siano conformi ai concetti che li descrivono e non semplicemente ad una semplice misura della loro distanza.

3.1.2 Obiettivo degli algoritmi di clustering

L'obiettivo degli algoritmi di clustering, è quindi di determinare il raggruppamento intrinseco di un insieme di dati non classificati. Ma come facciamo a decidere cosa costituisce un buon raggruppamento? Può essere dimostrato che non esiste un criterio che sia in assoluto il migliore e che sia indipendente dallo scopo che ci siamo prefissi. Conseguentemente, è l'utente che applica l'algoritmo che deve stabilire quale sia il criterio che più si adatti allo specifico scopo, in modo che i risultati possano soddisfare i requisiti dell'applicazione.

Ad esempio, possiamo essere interessati a trovare dei rappresentanti per dei gruppi di dati omogenei (data reduction), oppure nel trovare dei "gruppi naturali" e descrivere le loro proprietà sconosciute ("natural" data types), o ancora trovare gruppi utili ed adatti in determinati ambiti applicativi o trovare

oggetti di cui non sono note le proprietà (outlier detection, “rivelazione di dati erratici”).

3.1.3 Classificazione degli algoritmi di clustering

Gli algoritmi di clustering possono essere divisi nelle 4 seguenti categorie:

- Exclusive Clustering
- Overlapping Clustering
- Hierarchical Clustering
- Probabilistic Clustering

Nel primo caso i dati sono raggruppati in modo esclusivo, così se un certo dato appartiene ad un determinato cluster allora esso non potrà appartenere ad un altro cluster.

Il secondo tipo a differenza del primo usa delle regole fuzzy per raggruppare i dati, cosicché ogni dato può appartenere a due o più cluster con un differente grado di appartenenza.

Il cluster gerarchico, invece, si basa sulla fusione dei due cluster più “vicini” in modo iterativo. La condizione iniziale imposta, è quella di considerare ogni dato dell’insieme come un cluster a se stante. Dopo poche iterazioni viene quindi raggiunto il raggruppamento desiderato.

Infine l'ultimo tipo di clustering, fa uso di un approccio completamente probabilistico.

3.1.4 Misura della distanza

Una delle componenti più importanti per gran parte degli algoritmi di clustering è la misura della distanza tra gli oggetti da raggruppare. Se i dati in esame sono espressi tutti nella stessa unità di misura, allora la semplice distanza Euclidea è sufficiente per creare gruppi di dati simili. D'altronde anche in questo caso l'uso di questa misura, può essere fuorviante. Infatti, l'uso di differenti fattori di scala nell'esprimere i dati può portare a differenti raggruppamenti.

Minkowski Metric

Per dati con una dimensionalità elevata, una misura molto usata è la metrica di Minkowski, definita dalla legge:

$$d_P(x_i, x_j) = \left(\sum_{K=1}^{d} | x_{i,k} - x_{j,k} |^P \right)^{\frac{1}{P}}$$

nella quale *d* rappresenta la dimensione del dato. La distanza Euclidea è un caso speciale di questa metrica, in cui $p = 2$, mentre la metrica di Manhattan è il caso di *p* pari ad uno.

Comunque non esistono regole che ci permettono di scegliere una determinata misura per una certa applicazione. È il caso in cui le componenti dei vettori delle caratteristiche dei dati non sono direttamente comparabili. Può accadere ad esempio che le componenti non siano variabili continue ma categorie nominali, come i giorni della settimana. In questo caso, è solo il dominio dell'applicazione che ci può suggerire quale sia la formula appropriata per una determinata misura.

3.1.5 Applicazioni

Molti campi della scienza, come l'ingegneria, la zoologia, la medicina, la linguistica, l'antropologia, la psicologia ed il marketing, hanno contribuito allo sviluppo ed all'applicazione delle tecniche di clustering. Ad esempio, l'analisi dei cluster può aiutarci nel creare una cura "equilibrata" ed i relativi gruppi di controllo per uno studio pianificato. Se si scopre che ogni cluster contiene grosso modo un numero di cure corrispondenti ai soggetti sotto controllo, allora le differenze statistiche trovate tra i gruppi possono essere attribuite all'esperimento e non ad un'iniziale differenza tra i gruppi stessi.

Gli algoritmi di clustering sono usati concretamente e con successo in molti ambiti, ad esempio:

- marketing: per trovare gruppi di clienti con caratteristiche comportamentali simili, ricavate da un grande database che contiene le loro preferenze e gli acquisti passati
- biologia: classificazione di piante ed animali in base ai loro tratti biologici
- bibliografia: ordinamento intelligente di libri
- assicurazioni: identificare gruppi di soggetti per creare politiche di assicurazioni sui veicoli personalizzate; identificazione delle frodi
- urbanistica: identificare gruppi di abitazioni secondo il tipo, il valore commerciale e la locazione geografica
- sismologia: raggruppamento degli epicentri dei terremoti osservarti, per identificare zone a rischio sismico
- World Wide Web: classificazione di documenti; raggruppamento dei dati dei weblog per scoprire gruppi con simili pattern di accesso. Il clustering viene anche usato da alcuni motori di ricerca per determinare il contesto delle parole chiave della ricerca

I principali e più usati algoritmi di clustering sono i seguenti:

- Conceptual Clustering
- Hierarchical Clustering
- Subtractive Clustering
- Fuzzy C-Means Clustering
- Mixture of Gaussians
- K-means Clustering

Data la notevole importanza di questi algoritmi verranno brevemente descritti di seguito.

3.2 Conceptual Clustering

Le tecniche di clustering comuni hanno lo svantaggio di non fornire una descrizione intenzionale dei cluster ottenuti. Le tecniche di clustering concettuale invece provvedono a fornire tale descrizione ma è ben noto che tali algoritmi sono molto costosi in termini computazionali.

Prima di tutto quindi, viene applicato un algoritmo di clustering (non concettuale), ad esempio il K-means, in modo da diminuire la dimensione del problema. Viene, quindi applicato un algoritmo di clustering di tipo concettuale (nel nostro caso il Formal Concept Analysis) ai gruppi ottenuti nel

passo precedente. Quest'ultimo fornisce una descrizione intenzionale dei gruppi creati; ed è abbastanza efficiente, se il numero dei cluster scelti non è molto alto. Il concetto generico può essere quindi ricavato usando la tecnica del Formal Concept Analysis[5].

Il problema da risolvere è quindi il seguente: dato un insieme di documenti ed un lessico, si vuole fornire un raggruppamento dei documenti con ragionevole performance, che viene accompagnato da una descrizione intenzionale dei cluster.

Un tipico algoritmo di clustering usato nella pratica è il formal concept analysis (FCA). L'FCA si basa sulla descrizione matematica di 'concetto', ma per comprendere meglio tale definizione bisogna introdurre prima la nozione di '*contesto formale*':

Un *contesto formale* è una tripla K = (G,M,I), dove G è un insieme di oggetti, M è un insieme di attributi, ed I è un relazione binaria tra G ed M, che indica se G è caratterizzato o meno da M.

Da un contesto formale può essere derivato una gerarchia di concetti chiamata *concept lattice*:

$$\text{Dato } A \subseteq G, \text{ si definisce } A' := \{m \in M \mid \forall g \in A : (g,m) \in I\},$$

$$\text{e } B \subseteq M, \text{ si definisce } B' := \{g \in G \mid \forall m \in B : (g,m) \in I\}$$

Un concetto formale di un contesto formale (G,M,I) è definito come una coppia (A,B) con $A \subseteq G$, $B \subseteq M$, A'=B e B'=A. Gli insiemi A e B sono chiamati l'"extent" e l'"intent" del concetto formale (A,B). La relazione concetto-figlio / concetto-padre è formalizzata da:

$(A_1,B_1) \leq (A_2,B_2) \Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow B_1 \supseteq B_2)$

L'insieme di tutti i concetti formali di un contesto K insieme all'ordine parziale $\leq$ spesso costituisce uno schema completo, chiamato *concept lattice* di K ed indicato con $\beta(K)$

3.3 Hierarchical Clustering

3.3.1 Generalità dell'algoritmo

Dato un set di N entità che devono essere raggruppate, e una matrice di distanza (o similarità) N*N, che definisce la distanza di ogni oggetto con un altro, Il processo base del raggruppamento gerarchico (definito da S.C. Johnson nel 1967) è il seguente:

1. Si inizia con l'associare un cluster ad ogni entità, così se abbiamo N oggetti, inizialmente avremo N cluster, ognuno dei quali contiene un

solo oggetto. Poniamo quindi la distanza (similarità) tra i cluster pari alla distanza tra gli oggetti che essi contengono.

2. Ricerchiamo, quindi, la coppia di cluster più "vicini" (più simili) e li uniamo in unico cluster, in questo modo il numero di cluster verrà ridotto di un'unità.
3. Calcoliamo la nuova distanza (similarità) tra il nuovo cluster ed ognuno dei vecchi cluster.
4. Si ripete quindi il passo 2 e 3 fin quando le entità sono raggruppate nel numero di cluster desiderato.

Il passo 3 può essere effettuato in modo differente: effettuando una distinzione tra 3 tipi di raggruppamenti: *single linkage*, *complete linkage*, *average linkage*.

Nel *single linkage* (anche chiamato metodo di connettività o metodo di minimo), consideriamo come distanza tra un cluster ed un altro, la più piccola distanza di un membro del primo cluster con qualsiasi altro membro del secondo cluster.

Nel *complete linkage* (anche chiamato metodo del diametro o metodo del massimo), consideriamo la distanza tra un cluster ed un altro come la massima distanza tra un membro del primo cluster ed uno qualsiasi dei membri del secondo cluster.

Infine nell'*average linkage*, consideriamo come distanza tra un cluster ed un altro, la distanza media tra i membri del primo cluster e quelli del secondo. Una variazione di questo sistema è il metodo UCLUS di R. D'Andrade (1978)[6] che usa come distanza la deviazione standard, che è più resistente agli errori rispetto alla distanza media

Questo tipo di clustering è chiamato *agglomerativo* in virtù del fatto che aggrega i cluster in modo iterativo. Esiste anche un clustering di tipo separativo che compie l'operazione inversa rispetto al precedente, ovvero a partire da un unico cluster, ne crea dei sottogruppi. Tale metodo però viene raramente usato nella pratica

3.3.2 Single-Linkage Clustering: l'algoritmo

L'algoritmo è realizzato nella forma di uno schema agglomerativo che cancella righe e colonne dalla matrice di prossimità man mano che i vecchi cluster sono fusi in quelli nuovi.

La matrice di prossimità (di dimensione $N*N$) è D = [d(i,j)]. Ad ogni cluster è assegnato un numero sequenziale 0,1,…,(n-1) ed L(k) è il livello del k-esimo gruppo. Un cluster con numero di sequenza *m* è denotato come *m* è la prossimità tra i cluster *r* ed *s* è denotata come *d [r,s]*.

L'algoritmo si suddivide nei seguenti passi:

1. Si inizia con gruppi disgiunti che hanno livello L(0) = 0 e numero di sequenza $m = 0$
2. Si cerca la coppia di cluster meno "diversi", che chiameremo *s* ed *r* rispettivamente, secondo la legge $d[r,s] = min\ d[i,j]$ che rappresenta la minima distanza tra tutte le coppie di cluster nel livello di raggruppamento corrente
3. Si incrementa quindi il numero di sequenza ($m = m + 1$). Si fondono quindi i cluster *r* ed *s* in un singolo cluster per formare il successivo insieme di raggruppamento. Si imposta il livello di raggruppamento ad $L(m) = d[(r),(s)]$
4. Si aggiorna quindi la matrice di prossimità *D*, cancellando le righe e le colonne corrispondenti ai cluster *r* ed *s* ed aggiungendo una riga ed una colonna al cluster appena formatosi. La prossimità tra il nuovo cluster, indicata con $d(r,s)$ ed il vecchio cluster *k* viene definita in questo modo:

$$d[k, (r,s)] = min\ d[k,r], d[k,s]$$

5. Se tutti gli oggetti sono nello stesso gruppo o se sono suddivisi nel numero di cluster prefissati l'algoritmo giunge a termine altrimenti si ritorna al passo 2

3.3.3 Problemi

Le principali debolezze degli algoritmi di clustering di tipo agglomerativo sono le seguenti:

- non sono facilmente scalabili: la complessità computazionale è di almeno $O(n^2)$, in cui n rappresenta il numero di oggetti
- non è possibile annullare le operazioni effettuate nell'iterazione precedente

3.4 Fuzzy C-Means Clustering

Il Fuzzy C-Means (FCM) è un metodo di clustering che permette ad un sottoinsieme dei dati di appartenere a due o più cluster. Questo metodo (sviluppato da Dunn nel 1973 e migliorato da Bezdek nel 1981)[7] è frequentemente usato nel riconoscimento di pattern. Si basa fondamentalmente sulla minimizzazione della seguente funzione:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m || x_i - c_j ||^2$$

in cui m è un numero reale qualsiasi maggiore di 1, u_{ij} è il grado di appartenenza di x_i con il cluster j, x_i è la i-esima componente dei dati misurati di dimensione d, c_j è il centro di dimensione d del cluster, ed infine ||*|| è una norma che esprime la similarità o distanza tra i dati misurati ed il centro del

cluster. Il partizionamento fuzzy è realizzato attraverso un'ottimizzazione iterativa della funzione oggetto mostrata sopra, in cui viene ricorsivamente sostituita l'appartenenza al cluster u_{ij} ed il centro del cluster c_j tramite le seguenti leggi:

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{i=1}^{N} u_{ij}^m} \qquad u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|} \right)^{\frac{2}{m-1}}}$$

L'algoritmo convergerà quando $\max_{ij} \left\{ \left| u_{ij}^{(k+1)} - u_{ij}^{(k)} \right| \right\} < \varepsilon$, in cui ε è un valore che indica il limite entro cui l'algoritmo deve fermarsi, questo valore è compreso tra 0 ed 1, k indica il numero di iterazioni. Questa procedura converge ad un minimo locale o ad un punto di sella di J_m.

3.5 Mixture of Gaussians

3.5.1 Introduzione agli algoritmi di clustering basati su modello

C'è un altro sistema per trattare i problemi di clustering, un approccio basato su *modello*, che consiste nell'uso di un certo schema per i cluster, cercando di ottimizzare l'adattamento dei dati con il modello.

In pratica ogni cluster può essere matematicamente rappresentato da una distribuzione parametrica: Gaussiana (nel caso continuo) o una di Poisson (nel caso discreto). L'intero insieme dei dati è quindi modellato da una *composizione* di queste distribuzioni. Una singola distribuzione usata per modellare uno specifico cluster viene anche detta *componente* della distribuzione.

Il modello possiede con molta probabilità le seguenti caratteristiche:

- le distribuzioni componenti hanno dei "picchi" elevati (i dati in un cluster sono ravvicinati);
- il modello composto "copre" bene i dati (i pattern dominanti dei dati sono modellati dalle distribuzioni componenti)

I vantaggi principali del clustering basato sul modello sono:

- disponibilità di tecniche di inferenza statistica ben conosciute
- flessibilità nella scelta della distruzione componente
- ottenere una stima della densità per ogni cluster
- disponiamo di una classificazione flessibile

3.5.2 L'algoritmo Mixture of Gaussians

Il metodo di clustering di questo tipo più ampiamente usato, è quello basato sull'apprendimento di una *composizione di Gaussiane:* in realtà possiamo considerare i cluster come distribuzioni Gaussiane centrate nel proprio baricentro, come è possibile vedere nella figura, nella quale i cerchi grigi rappresentano la prima variazione della distribuzione:

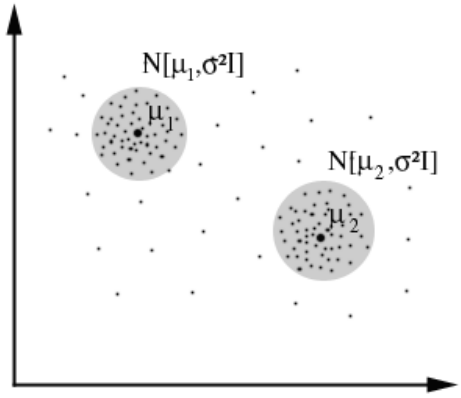


Figura 2 - Clustering di dati distribuiti secondo una Gaussiana

L'algoritmo opera nel seguente modo:

- si seleziona in modo casuale la componente (la distribuzione Gaussiana) con probabilità $P(\omega_i)$
- si campiona un punto $N(\mu_i, \sigma^2 I)$

Supponiamo di avere:

- $x_1, x_2,..., x_N$

- $P(\omega_1), \ldots, P(\omega_k), \sigma$

Possiamo ottenere la probabilità dei campioni da: $P(x \mid \omega_i, \mu_1, \mu_2, \ldots, \mu_k)$. Quello che vogliamo massimizzare è $P(x \mid \mu_1, \mu_2, \ldots, \mu_k)$ (probabilità di un dato, noto il centro della Gaussiana)

$$P(x \mid \mu_i) = \sum P(\omega_i) P(x \mid \omega_i, \mu_1, \mu_1, \ldots, \mu_k)$$

è la base iniziale per scrivere la funzione di probabilità:

$$P(dati \mid \mu_i) = \prod_{i=1}^{N} \sum P(\omega_i) P(x \mid \omega_i, \mu_1, \mu_1, \ldots, \mu_k)$$

Adesso occorrerebbe calcolare la funzione di probabilità, calcolando $\frac{\partial L}{\partial \mu_i} = 0$, ma che risulterebbe troppo complessa. Questo è il motivo per cui usiamo un algoritmo semplificato chiamato EM (Expectation-Maximization, Dempster, Laird and Rubin, 1977). Tale algoritmo è quello che viene usato nella pratica per trovare la composizione di Gaussiane che riescono a modellare il set di dati da clusterizzare.

3.6 K-means Clustering

Il K-means è un algoritmo per il partizionamento (o clustering) di un numero N di dati in numero K di sottoinsiemi disgiunti, detti S_j che contengono un numero N_j di dati così da minimizzare il criterio della somma quadratica

$$J=\sum_{j=1}^{K}\sum_{n\in S_j}|x_n-u_j|^2$$

in cui x_n è un vettore che rappresenta l'n-esimo oggetto ed μ_j il centroide geometrico dei dati nel sottoinsieme S_j.

In generale, l'algoritmo non raggiunge un minimo globale di J. Infatti, poiché l'algoritmo usa un'assegnazione discreta più che un insieme di parametri continui, il minimo raggiunto non può essere propriamente definito un minimo locale. Malgrado queste limitazioni, l'algoritmo viene frequentemente usato con soddisfacenti risultati, grazie anche alla facilità della sua implementazione.

Il K-means (MacQueen, 1967)[8] è uno dei più semplici algoritmi di apprendimento non supervisionato che risolve problemi noti di clustering. La procedura segue un metodo semplice e facile per classificare un prefissato insieme di dati in un numero di cluster noto a priori (che chiameremo k). L'idea principale è di definire k centroidi, uno per ogni cluster. Tali centroidi, dovrebbero essere posizionati in modo astuto, poiché a disposizioni diverse

corrispondo risultati differenti. Quindi la scelta migliore è di posizionare i vari centroidi il più possibile lontanamente tra loro.

Il passo successivo, consiste nel prendere tutti i dati appartenenti ad un certo insieme e di assegnarli al centroide più vicino. Il primo passo è completato quando tutti i punti vengono elaborati e viene creato un raggruppamento iniziale. A questo punto è necessario ricalcolare *k* centroidi nuovi come baricentro dei cluster ottenuti al passo precedente. Dopo che questi nuovi centroidi sono stati calcolati, viene effettuata una nuova assegnazione tra i dati ed i nuovi centroidi, sempre in modo che la distanza sia minima. Viene instaurato un ciclo; come risultato, possiamo notare che i centroidi cambiano la loro posizione ad ogni iterazione fin quando la non vengono più effettuati dei cambiamenti. In parole povere i centroidi non si "muovono" ulteriormente. Per finire, questo algoritmo mira alla minimizzazione di una funzione oggetto, nel nostro caso di una funzione di errore quadratico. La funzione oggetto:

$$J = \sum_{j=1}^{K} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

in cui $\left\| x_i^{(j)} - c_j \right\|^2$ è una determinata misura della distanza tra un dato $x_i^{(j)}$ ed il centro del cluster c_j, è un indicatore della distanza degli *n* punti dal loro relativo centroide.

L'algoritmo si suddivide nei seguenti passi:

1. si posizionano i *K* punti nello spazio degli oggetti che devono essere clusterizzati. Tali punti rappresentano i centroidi dei gruppi iniziali
2. si assegna ogni oggetto al gruppo il cui centroide è più vicino all'oggetto stesso
3. quando tutti gli oggetti sono stati assegnati, si ricalcola la posizione dei *K* centroidi
4. si ripetono i passi 2 e 3 fin quando i centroidi non si spostano più. Questo processo produce una separazione degli oggetti in gruppi di cui è possibile calcolare la metrica da minimizzare

Sebbene è possibile provare che la procedura giunge a termine, l'algoritmo K-means non troverà necessariamente la soluzione ottimale, che corrisponde al minimo globale della funzione oggetto.

L'algoritmo è significativamente sensibile alla disposizione dei centri dei cluster che generalmente sono scelti in modo casuale. Per ovviare a questo problema il K-means può essere eseguito varie volte per ridurre l'effetto della disposizione casuale dei centroidi.

CAPITOLO IV

Spazi di Colore

4.1 Lo spazio RGB

RGB è il nome di un modello di colori additivo che si basa sui tre colori primari: Rosso (Red), Verde (Green) e Blu (Blue), da cui appunto il nome RGB. Questo modello viene usato nel digitale per trasmettere immagini a colori.

Un'immagine può infatti essere scomposta, attraverso filtri o altre tecniche in questi colori base che, miscelati tra loro danno quasi tutto lo spettro dei colori visibili, con l'eccezione delle porpore. Il bianco è ottenuto con un mix di circa lo 0.7/0.2/0.1 dei canali verde (G), rosso (R) e blu (B).

L'RGB è un modello additivo: unendo i tre colori con la loro intensità massima si ottiene il bianco (tutta la luce viene riflessa). La combinazione delle coppie di colori dà il ciano, il magenta e il giallo.

Un colore nel modello RGB può essere descritto indicando la quantità di rosso, verde e blu. Ognuno di questi può variare tra un minimo (colore assente) ed un massimo (piena intensità). Se tutti i colori hanno il valore minimo il colore risultante sarà nero, se invece tutti i colori hanno il valore massimo allora il risultato sarà il colore bianco. Un aspetto del modello RGB che può portare a confusione è che i colori possono essere rappresentati secondo differenti notazioni.

La scienza dei colori dà ad essi una variazione nell'intervallo tra 0.0 (minimo) ed 1.0 (massimo). La maggior parte delle formule relative ai colori fa uso di questi valori detti normalizzati.

Il valore di un certo colore può anche essere scritto come percentuale, da 0% (minimo) a 100% (massimo). Per convertire dalla notazione precedente a questa basta ovviamente moltiplicare per cento.

I valori dei colori possono anche essere scritti come numeri che variano nell'intervallo [0:255], e ciò avviene moltiplicando per 255 la notazione scientifica vista in precedenza. Questa notazione è comunemente usata in ambito informatico poiché è semplice rappresentare ogni componente di un colore come 1 byte = 8 bit = 2^8 = intervallo 0:255.

Questo stesso range viene talvolta scritto in notazione esadecimale, con anteposto un prefisso '#' per indicare appunto tale base. Questa notazione è conveniente in quanto permette la rappresentazione di ogni componente con un formato fisso a due cifre. Questa convenzione è spesso usata per indicare i colori nelle pagine web.

Lo spazio RGB è ognuno degli spazi di colori di tipo additivo basato sul modello RGB. L'RGB è un modello di colore conveniente per la computer graphics poiché il sistema visivo degli uomini funziona in un modo simile ma

non identico ad uno spazio di colori RGB. La maggior parte degli spazi di colore RGB standard usati sono l'sRGB e l'Adobe RGB[9].

La rappresentazione grafica dello spazio RGB può essere quella di tre assi cartesiani. Ognuno dei colori può essere rappresentato nel cubo delimitato dall'origine degli assi al punto (1.0,1.0,1.0). all'interno troveremo tutti i possibili colori rappresentabili con questo spazio di colore. Ogni colore viene quindi posizionato associando ad ognuno degli assi uno dei tre colori primari e viene individuato grazie alle coordinate del colore stesso riportate nello spazio RGB

4.2 Lo spazio HSV

Il modello HSV (Hue, Saturation, Value), anche conosciuto come HSB (Hue, Saturation, Brightness), definisce uno spazio di colori in termini di tre componenti:

- Hue, il tipo di colore (come rosso, blu, o giallo):

  varia da 0 a 360 (o come avviene in molte applicazioni da 0% a 100% considerando la normalizzazione)
- Saturation, la intensità del colore:

  varia da 0% a 100%

Anche conosciuta come “purezza” dall'analogia con la purezza dell’eccitazione delle quantità colorimetriche e la purezza colorimetrica.

Minore è la saturazione di un colore, maggiore sarà la sua tendenza al grigio e saranno quindi visibili i colori più sbiaditi, risulta utile definire la “desaturazione” come funzione inversa rispetto alla saturazione

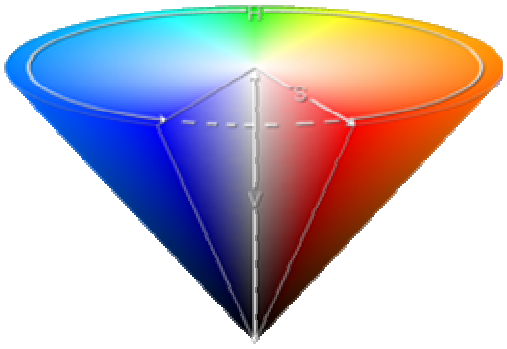
- Value, la luminosità di un colore:

  varia tra 0% e 100%

Il modello HSV fu creato nel 1978 da Alvy Ray Smith. è una trasformazione non lineare dello spazio di colori RGB, e può essere usata nelle progressioni di colori.

Il modello HSV è comunemente usato nella applicazioni grafiche. Una sua rappresentazione può essere quella di una ruota colorata. In essa la colorazione è rappresentata da una da regione circolare; una regione triangolare separata dalla prima può essere usata per rappresentare la saturazione e la luminosità. Tipicamente, l'asse verticale del triangolo indica la saturazione, mentre l'asse orizzontale corrisponde alla luminosità. In questo modo, un colore può essere scelto decidendo prima la tonalità e quindi selezionando la saturazione e luminosità desiderate.

Un altro sistema per visualizzare il modello HSV è il cono.

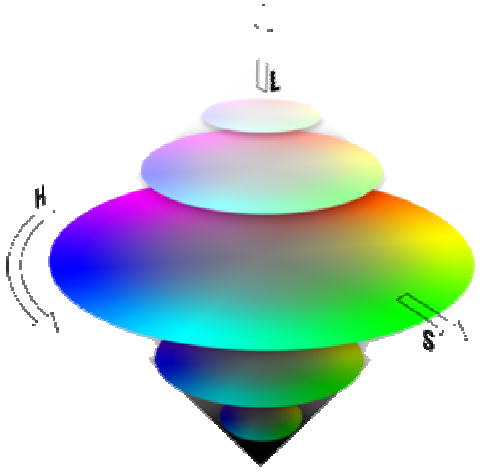


Rappresentazione Spazio HSV

In questa rappresentazione, la tonalità è graficata come una forma conica tridimensionale della ruota colorata. La saturazione è rappresentata dalla distanza dal centro dell'asse del cono mentre la luminosità è la distanza dalla punta del cono. Alcune rappresentazione usano un cono a base esagonale, invece di un cono a base circolare. Questo metodo si presta meglio alla visualizzazione dell'intero spazio HSV in un oggetto singolo; d'altronde a causa della sua natura tridimensionale non si presta bene alla selezione dei colori nei computer che si basa su un'interfaccia bidimensionale.

4.3 Lo spazio HSL

L'acronimo di spazio HSL, anche chiamato HSI, sta per spazio Hue Saturation Lightness (o Intensity). Mentre l'HSV può essere rappresentato da un cono colorato, l'HSL è graficato come un doppio cono con le basi unite. Entrambi i modelli sono deformazioni non lineari del cubo di colori



Rappresentazione spazio HSL

RGB. I due apici del doppio cono HSL corrispondono al nero ed al bianco. Il parametro angolare corrisponde alla colorazione, mentre la distanza dall'asse mediano corrisponde alla saturazione, ed infine la distanza lungo l'asse mediano (congiungente bianco e nero) corrisponde all'intensità luminosa.

L'HSL è simile all'HSV ma riflette meglio la nozione di saturazione e luminosità come due parametri indipendenti.

Nell'HSL, la componente della saturazione va sempre da un colore completamente saturo all'equivalente grigio con saturazione nulla.

La luminosità nell'HSL copre l'intero range che va dal bianco al nero attraverso una determinata colorazione.

4.4 Il Cromaticity Space

Il colore di un'immagine acquisita da un qualunque dispositivo digitale è rappresentato da 3 componenti: rosso, verde e blue. La combinazione di queste componenti genera uno spazio tridimensionale chiamato "spazio RGB". Studi fisici mostrano che i colori di superfici dielettriche giacciono su un volume ristretto dello spazio RGB, che ha una forma cilindrica con raggio molto piccolo.

Quando l'intensità luminosa cambia, i colori di una superficie si muovono lungo una linea coincidente con l'asse del cilindro suddetto.

I colori di una di queste superfici si dispongono al variare della luminosità in una nuvola di punti di forma allungata come previsto dagli studi fisici. Come previsto dalla teoria al variare della sorgente luminosa e dell'intensità la regione di colori rilevata si sposterà lungo l'asse ma sempre rimanendo contenuta nella sezione del cilindro menzionato prima. Inoltre usando il cromaticity space è possibile ridurre il numero di componenti necessarie a rappresentare il modello di colori di una certa superficie o oggetto proiettando

un generico punto dello spazio RGB sul piano perpendicolare all'asse principale della nube individuata. Chiameremo quindi questo piano "cromaticity plane". È da notare che tutti i possibili colori dello spazio RGB saranno proiettati su una regione piana di forma triangolare che verrà detta "cromaticity space"[10].

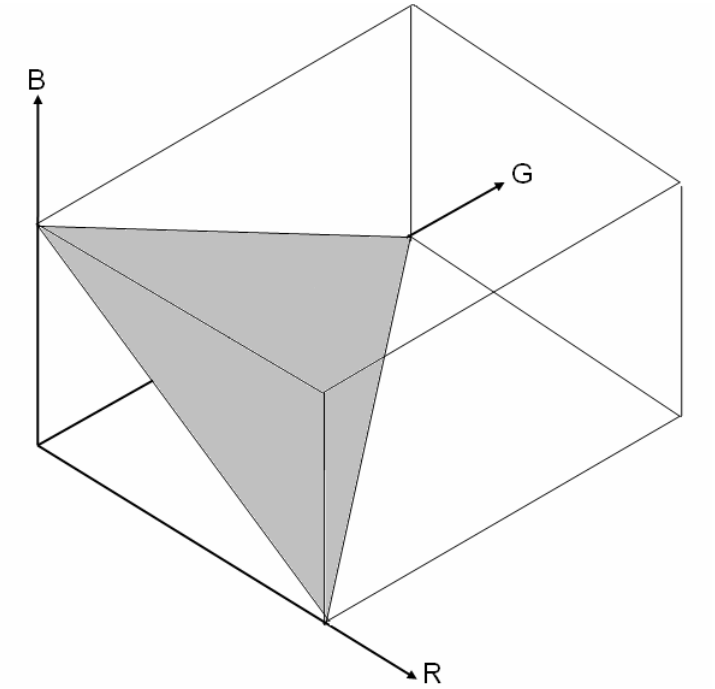


Figura 3 - Spazio RGB e piano s,t in grigio chiaro

Questo piano ci permette una rappresentazione basata solo sui colori prescindendo dalla sorgente luminosa cui essi sono sottoposti. Nella pratica se si vuole costruire un modello basato sui colori di un oggetto, tale rappresentazione ci permetterà di creare un modello che è indipendente dalla

condizioni atmosferiche. Questa spazio infatti è invariante alle condizioni luminose: l'area individuata nello spazio della cromaticità è sempre simile a prescindere da illuminazione artificiale o naturale, tempo sereno o nuvoloso oppure momento della giornata in cui viene ripresa un'immagine.

Se vogliamo effettuare la trasformazione di un generico colore nello spazio RGB nel corrispondente nel cromaticity space ([R,G,B] → [s,t]), dovremo usare le seguenti leggi:

$$s = \alpha \frac{2R - G - B}{R + G + B} \qquad t = \beta \frac{G - B}{R + G + B}$$

in cui α e β sono fattori di scala che dipendono dall'applicazione.

CAPITOLO V

Sistema Implementato

5.1 Introduzione

Il sistema software implementato ha l'obiettivo di determinare se un'immagine proveniente da una qualunque fonte video, come una webcam, una videocamera di sorveglianza, un'immagine statica oppure una sequenza di fotogrammi provenienti da un video, contiene un'insieme di pixel che rappresentino un incendio.

Per raggiungere quindi lo scopo si è fatto uso dell'ambiente di sviluppo Matlab 6.5 che offre tutti gli strumenti necessari all'elaborazione di immagini e matrici in modo molto semplice.

Il sistema sviluppato si base fondamentalmente sulla creazione di un *modello di un fuoco* partendo da un insieme di immagini contenenti degli incendi. Tale set di immagini può essere scelto o in maniera generica, ovvero usando immagini provenienti da varie fonti, ad esempio effettuando una ricerca sul web e scaricando delle immagini, oppure, nel caso in cui sia possibile, usando delle immagini dell'ambiente in cui il sistema andrà ad operare, generando degli incendi controllati. Ovviamente usando il secondo metodo il sistema risulterà più affidabile; in particolar modo il numero di falsi positivi, ovvero il numero di immagini in cui viene erroneamente rilevato un incendio, diminuirà sensibilmente.

Una volta creato il modello, il software sarà pronto ad operare: le immagini provenienti dalla fonte video di controllo, saranno processate per determinare l'eventuale presenza di fuoco.

Durante la fase di test, il processo di elaborazione delle immagini ricevute all'ingresso del sistema si scompone nei blocchi evidenziati nel seguente schema

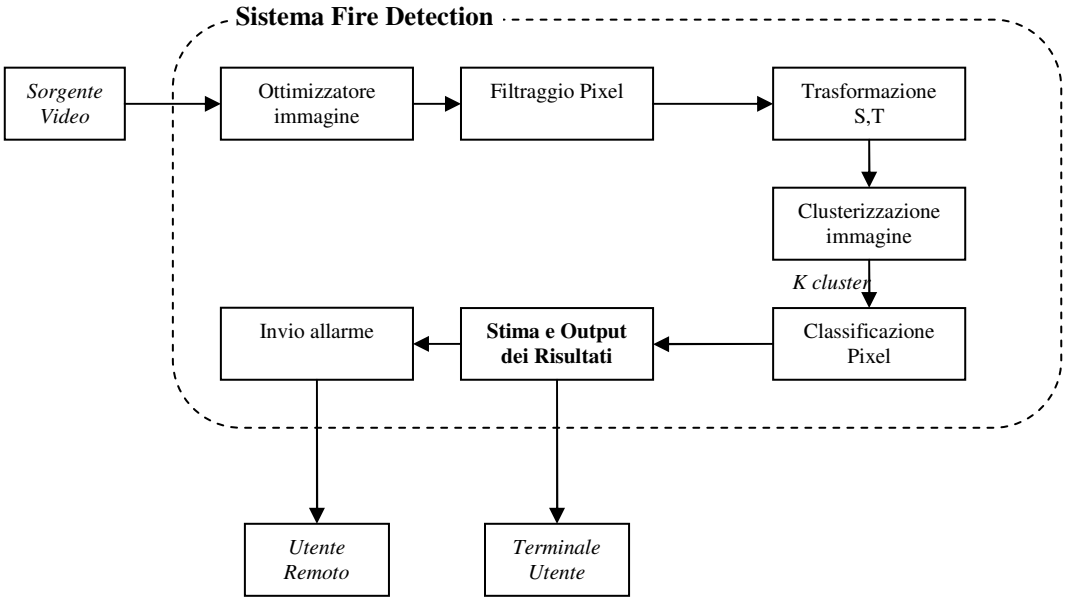


Figura 4 - Schema a blocchi del sistema implementato

L'ingresso è la sorgente video che deve fornire un certo numero di fotogrammi per unità di tempo ad una determinata risoluzione. Mentre è

possibile variare la risoluzione video delle immagini (il sistema è in grado di lavorare con qualunque risoluzione), la frequenza dei fotogrammi è fissa ed è calcolata sulla base delle caratteristiche hardware su cui girerà il sistema.

L'uscita associata al blocco *Terminale Utente* rappresenta tutti i messaggi di funzionamento e di allarme che vengono visualizzati sullo *standard output* della macchina che esegue il software. Inoltre, è stato utile inserire istruzioni che creino dei file di log compatibili con un qualunque programma di calcolo (formato di testo CSV), per poter eseguire delle stime sia sulle performance che sull'accuratezza del programma.

La seconda uscita denominata *Utente Remoto* rappresenta i messaggi d'allarme e le relative descrizioni che vengono inviate ad un utente o ad un sistema remoto. Nel software realizzato, allo scattare di un allarme viene inviato un messaggio di posta elettronica ad uno o più utenti. Ovviamente questo blocco può essere esteso andando a realizzare sistemi di notifica verso altre macchine tramite una connessione TCP/IP; oppure può essere realizzato un sistema che si interfaccia con un gateway della rete GSM per inviare un SMS verso un numero cellulare.

5.2 Creazione del modello

5.2.1 Funzionamento

La creazione del modello è la prima e più delicata fase di tutto il processo. Come già spiegato nell'introduzione, questo passo potrebbe essere evitato, facendo uso di un modello generico di incendio, che viene preventivamente creato usando delle immagini provenienti da varie fonti e che quindi ricopre grossolanamente la maggioranza delle tipologie di incendi e ambienti in cui il sistema andrà ad operare.

Affinché, però, il sistema possa funzionare al meglio, riducendo così i rilevamenti erronei ed in particolar modo il numero di falsi positivi, il sistema deve essere tarato in base dell'ambiente operativo. La taratura avviene creando un modello specifico, che è ottenuto usando come campioni le foto relative all'ambiente di funzionamento.

La procedura di creazione del modello richiederà all'utente il numero di immagini che dovranno essere usate. Iterativamente, per il numero di immagini specificato in precedenza, si richiede di indicare l'immagine da caricare e successivamente viene aperta una finestra con la foto caricata (fig. 5).

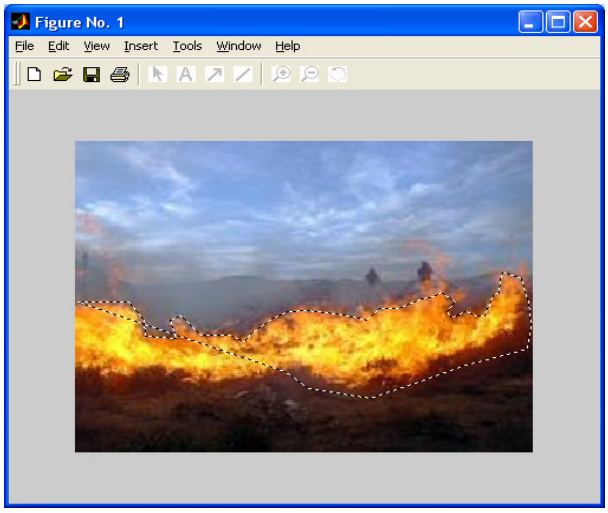


Figura 5 - Finestra per la selezione dell'area dell'incendio

Qui andrà selezionato con una poligonale chiusa l'area che racchiude il fuoco in modo da rendere il modello più preciso.

Alla fine del processo, viene creato il modello dell'incendio all'interno di una matrice bidimensionale che verrà quindi memorizzato per eseguire i confronti con le immagini da analizzare.

5.2.2 Selezione dei campioni

La selezione del set di immagini per creare il modello è di importanza cruciale per il buon funzionamento del sistema. Il primo parametro che deve essere indicato durante questa fase è il numero di campioni che andranno a costituire il modello. Ovviamente ad un numero maggiore corrisponde una maggiore precisione del modello stesso, infatti quest'ultimo è realizzato

tramite un'analisi statistica dei pixel. D'altronde un numero di campioni troppo elevato, renderebbe troppo tedioso il processo di creazione del modello, in quanto per ogni immagine bisognerà selezionare l'area interessata dall'incendio. D'altro canto, questo parametro non può essere troppo piccolo perché altrimenti verrebbe prodotto un modello impreciso. Un valore ottimale per questo parametro è quindi un numero che viene scelto empiricamente.

Oltre al numero di campioni, risulta importante la qualità delle immagini scelte per il modello. In linea generale andranno selezionate delle foto di incendi con la giusta esposizione, che non siano "sbiadite" ovvero affette dal fenomeno dello smoothing dovuto ad una bassa banda passante dei sensori CCD.

Una volta scelto un dispositivo che rispetti i criteri qualitativi sopraindicati, risulta importante la posizione del dispositivo rispetto all'incendio e, nel caso di un ambiente con illuminazione naturale, il momento della giornata durante il quale saranno riprese le istantanee.

La posizione della sorgente video dovrà essere scelta in modo che essa vada a ricoprire l'area da monitorare e possibilmente in modo che non sia né troppo distante, né troppo vicina all'incendio. Infatti nel primo caso potrebbe accadere che l'incendio ricopra solo una piccola zona dell'immagine rilevata (fig. 8) ed in questo caso il sistema non riesce a "distinguere" l'incendio dal

background. Nel secondo caso, invece, la foto potrebbe essere occupata per la maggior parte dal nucleo delle fiamme (fig. 9), che a causa dell'alta luminosità comprometterebbe la variazione cromatica dei pixel adiacenti.

Figura 6 - Esempio di immagine ben scattata

Figura 7 - Esempio di immagine ben scattata

Figura 8 - Esempio di immagine con fuoco troppo distante

Figura 9 - Esempio di immagine con fuoco troppo ravvicinato: tutti i pixel tendono al rosso

Inoltre affinché il modello sia accurato è importante che le immagini campione siano scattate in diversi momenti della giornata in modo che il modello possa inglobare le variazioni cromatiche rispetto alla luce naturale. Si

è potuto constatare che 3 gradi di illuminazione sono sufficienti per rendere il modello immune alla variazione della luce (questo grazie anche alla trasformazione s,t). In linea di massima basta scegliere il primo mattino o il tardo pomeriggio, mezzogiorno ed infine in tarda serata o di notte.

5.2.3 Trasformazione delle immagini

Una volta scelto un campione ed applicata la maschera di selezione dell'incendio, i pixel segnati come 'fuoco', vengono passati ad una funzione che ci permette di passare dallo spazio tridimensionale (R,G,B) al piano bidimensionale (s,t). Questa trasformazione viene usata per diminuire la complessità computazionale del sistema e perché lo spazio della crominanza (s,t) permette di effettuare analisi sul colore prescindendo dalla luminosità.

Il passaggio dallo spazio RGB al piano (s,t) permette di diminuire notevolmente il tempo di elaborazione delle immagini, quando queste vengono confrontate col modello durante la fase di fire detection. La formula applicata infatti associa alle tre componenti R G e B, due componenti s e t.

Inoltre tale trasformazione permette un'analisi sul colore dei pixel che prescinde dalla loro luminosità: in pratica due istantanee scattate in diversi momenti della giornata hanno grafici nel piano s,t simili. La loro distribuzione nel piano è simile e differisce solo per la scala, quindi è possibile comparare

le immagini con un modello di fuoco indipendente dalla luminosità ambientale.

5.2.4 Creazione della matrice modello

Una volta applicata la maschera di selezione e calcolata la trasformazione dei pixel nello spazio della crominanza, i dati rilevati dalle varie immagini vanno quindi inseriti in una matrice che rappresenterà il nostro modello dell'incendio. Tale matrice verrà indicata da questo momento in poi come *matrice_modello*.

Quest'ultima rispecchierà con gli opportuni accorgimenti, la distribuzione dei dati delle varie immagini nel piano della crominanza, ossia si farà in modo che le righe e le colonne della matrice ne rappresentino l'ascissa (*s*) e l'ordinata (*t*), ed ogni elemento di essa rappresenti il numero di volte in cui un certo pixel si presenta nelle immagini del modello.

Per riuscire nello scopo, prima di tutto è stato necessario assegnare a *matrice_modello* le opportune dimensioni, affinché possa contenere i dati all'uscita della trasformazione:

$$matrice\_modello[\ 3 * \alpha + 1,\ 2 * \beta + 1\ ]$$

si è scelto un numero di righe pari a 3 volte α in quanto la variabile *s* secondo la trasformazione indicata nel paragrafo precedente varia nel range [–α, 2α],

in modo del tutto analogo si è scelto un numero di colonne pari a 2 volte β perché la variabile *t* varia nel range [–β ,β]. In entrambi i casi si è aggiunta una riga ed una colonna per conformarci alla notazione dell'ambiente di sviluppo usato e per rendere i dati più comprensibili. Tale notazione non prevede l'uso di indici nulli per le matrici ed i vettori.

Una volta inizializzata la matrice del modello, vengono elaborati i dati relativi alla trasformazione dei pixel. Il primo passo effettuato è quello di arrotondare all'intero più vicino i valori di *s* e *t* relativi ad un certo pixel, questo passo è necessario poiché dovremo usare questi due valori come indici della matrice modello (che sono appunto degli interi). Secondariamente, sarà necessario traslare tali valori in modo che siano conformi alla notazione scelta in precedenza, quindi la variabile *s* e la variabile *t* subiranno le seguenti trasformazioni:

$$s = s + \alpha + 1 \qquad t = t + \beta + 1$$

in questo modo ogni pixel avrà una diretta corrispondenza nel piano della crominanza (s,t) rappresentato dalla matrice del modello.

Come ulteriore facilitazione prima di applicare la trasformazione vista in precedenza, viene effettuato un filtraggio relativo ai pixel "troppo scuri" (valori di Red, Green e Blue contemporaneamente inferiori a 30) e troppo

"chiari" (valori di Red, Green e Blue contemporaneamente superiori a 240) senza ledere in generalità.

L'obiettivo ultimo di questa fase è di dare un peso ad un certo pixel di fuoco all'interno del modello. Sono state scelte quindi due diverse soluzioni:

- un certo pixel (di fuoco) all'interno di una stessa immagine viene considerato una ed una sola volta, così se questo pixel compare ad esempio N volte esso verrà valutato come se fosse presente una volta. Ciò viene realizzato usando 2 vettori temporanei *s_v* e *t_v* (di dimensioni pari al numero di pixel totali) che contengono i valori di *s* e *t* per ogni pixel. Gli elementi di tali vettori vengono usati come indici per la matrice modello che verrà incrementata di uno ad ogni immagine
- ogni pixel (di fuoco) viene considerato per ogni volta che si presenta all'interno di ogni immagine. Se questo pixel detto *p* avrà, una volta applicata la trasformazione (s,t), s=s_p e t=t_p allora l'elemento di indici [s_p, t_p] all'interno di *matrice_modello* verrà incrementato di 1

A causa del rumore intrinseco dei dispositivi di acquisizione video, nel modello potrebbero comparire dei punti isolati da quelli che effettivamente rappresentano pixel associati al fuoco. Per risolvere questo problema, viene

applicato alla matrice del modello un filtro di tipo mediano che opera su un'area di 3x3 pixel

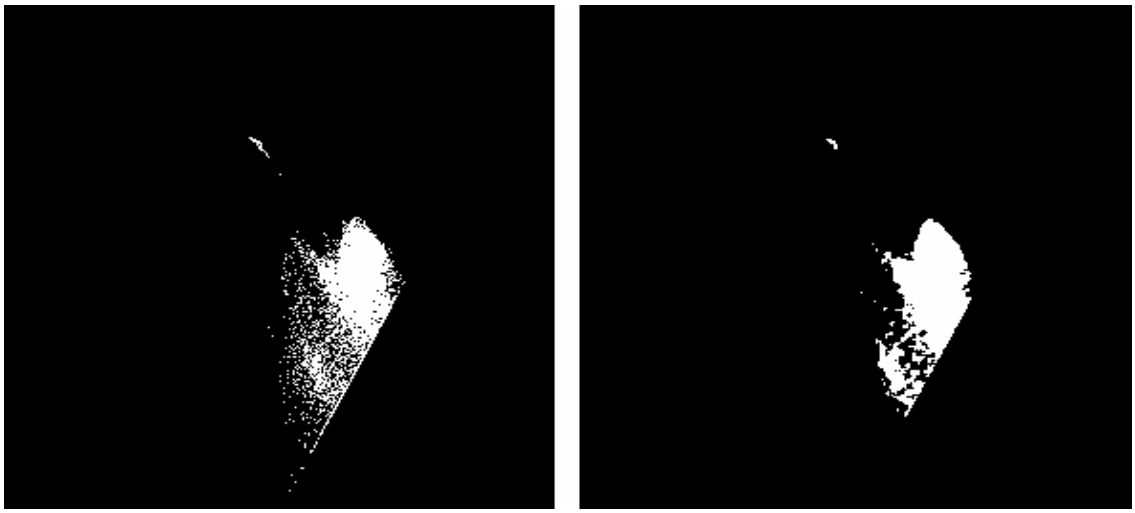


Figura 10 - Modello prima e dopo l'applicazione del filtro mediano

Come risultato sarà possibile notare (fig. 10) che tutti i pixel isolati vengono eliminati. Questi pixel corrispondono appunto ad errori dovuti a rumore o ad una selezione grossolana dell'area che racchiude il fuoco.

Infine la matrice modello viene normalizzata rispetto al massimo della matrice stessa

$$matrice\_modello_{ij} = matrice\_modello_{ij} / max(matrice\_modello)$$

in modo che tutti gli elementi della matrice ricadano nell'intervallo [0, 1] e quindi il modello possa essere rappresentato e visualizzato come un'immagine in scala di grigi (fig. 11).

Figura 11 - Modello prima e dopo la normalizzazione

In questo modo è possibile associare un significato al livello di grigio di ogni pixel del modello: al tendere di un punto verso il bianco, l'importanza del pixel stesso cresce proporzionalmente, ovvero un punto bianco indica una frequenza più alta rispetto a quella di un punto grigio scuro; i punti del modello neri indicano l'assenza dei corrispondenti pixel dal modello.

Poiché questo procedimento è un'analisi statistica sui pixel, nel modello verranno individuati anche pixel che ricadono nella selezione del fuoco per errore dell'utente poiché la selezione dell'area fuoco è generalmente molto grossolana. Questi pixel presenti solo sporadicamente avranno comunque un peso nel modello, per questo motivo vengono esclusi tutti quei campioni che nel grafico s,t hanno un valore molto basso. Viene effettuata un'operazione di

thresholding, ossia viene stabilita una soglia e tutti i punti del modello che non hanno un valore superiore a tale soglia vengono esclusi ponendoli a 0. Nel caso in questione è stato scelto un valore della soglia pari a 0.13.

5.3 Fase di test: Fire Detection

5.3.1 Ottimizzazione dell'immagine

La prima operazione da effettuare su un'immagine acquisita è di cercare di renderla quanto più possibile nitida, e contrastata, senza che essa sia troppo luminosa. Questi passi preliminari sono molto importanti in quanto permettono di eliminare, almeno in parte, i difetti intrinseci associati ai dispositivi di acquisizione video di qualità medio-bassa. Generalmente le webcam usate in un ambiente esterno illuminato da luce naturale presentano una forte luminosità che rende i colori sbiaditi, inoltre il contrasto dell'immagine è generalmente basso.

Per questo motivo, si è deciso di applicare una gamma correction, ovvero una relazione che si applica ad ogni pixel dell'immagine:

$$md\_px = 255\left(\frac{or\_px}{255}\right)^{\gamma}$$

in cui si impone $\gamma = 0.5$.

Inoltre su ognuno dei piani R, G e B si è deciso di applicare un'equalizzazione dell'istogramma. Grazie a queste operazioni si ottiene quindi un'immagine correttamente illuminata e con un contrasto che permette di evidenziare i dettagli dell'immagine.

Inoltre nei casi in cui si presentano rumori dovuti alla sorgente di acquisizione (ad esempio pixel di colore diverso rispetto a quelli che lo circondano), è stato applicato un filtro mediano che permette associa al colore di un pixel, la media dei colori dei pixel che lo circondano.

5.3.2 Filtraggio dei pixel

Una volta che l'immagine è stata migliorata dal punto di vista cromatico è possibile applicare un filtro sui pixel in modo da scartare quelli le cui proprietà non soddisfano determinati criteri.

Per la creazione di questo filtro ci si è basati sulle caratteristiche intrinseche del fuoco: colorazione ed un alto contrasto rispetto al background.

Per determinare le proprietà relative al colore ci si è basati su uno studio delle tre componenti cromatiche di ogni pixel: rosso, verde, blu.

Da un'analisi sperimentale si è potuto verificare in maniera visiva che le componenti assumevano determinati valori in prossimità di un incendio.

In particolare si nota che la componente rossa assume sempre un valore elevato, questa valutazione qualitativa è stata modellata imponendo che la componente rossa sia maggiore di 128 (assumendo che il rosso possa variare nel range [0…255]).

Una seconda restrizione è stata introdotta per la componente blu: questa componente si mantiene sempre molto bassa, escludendo i casi in cui i pixel considerati siano quelli appartenenti al nucleo dell'incendio. In questo caso infatti il colore sarà prossimo al bianco, caso in cui tutte e tre le componenti assumono valori prossimi a 255. Per escludere, quindi, i casi in cui la componente blu è elevata si è usata la legge per cui un pixel non deve avere componente blu inferiore a 128 e contemporaneamente componente rossa inferiore a 200.

Per quanto concerne la componente verde, invece, non si è potuta individuare alcuna caratteristica comune ai pixel di fuoco, per questo motivo non è stata introdotta alcuna restrizione sul verde.

Per determinare le proprietà relative al contrasto ed alla luminosità, si è prima applicata una trasformazione sull'immagine per portarla dal piano RGB al

piano HSV. In questo piano si è effettuata una valutazione qualitativa sulle componenti S (saturazione) e V (luminosità).

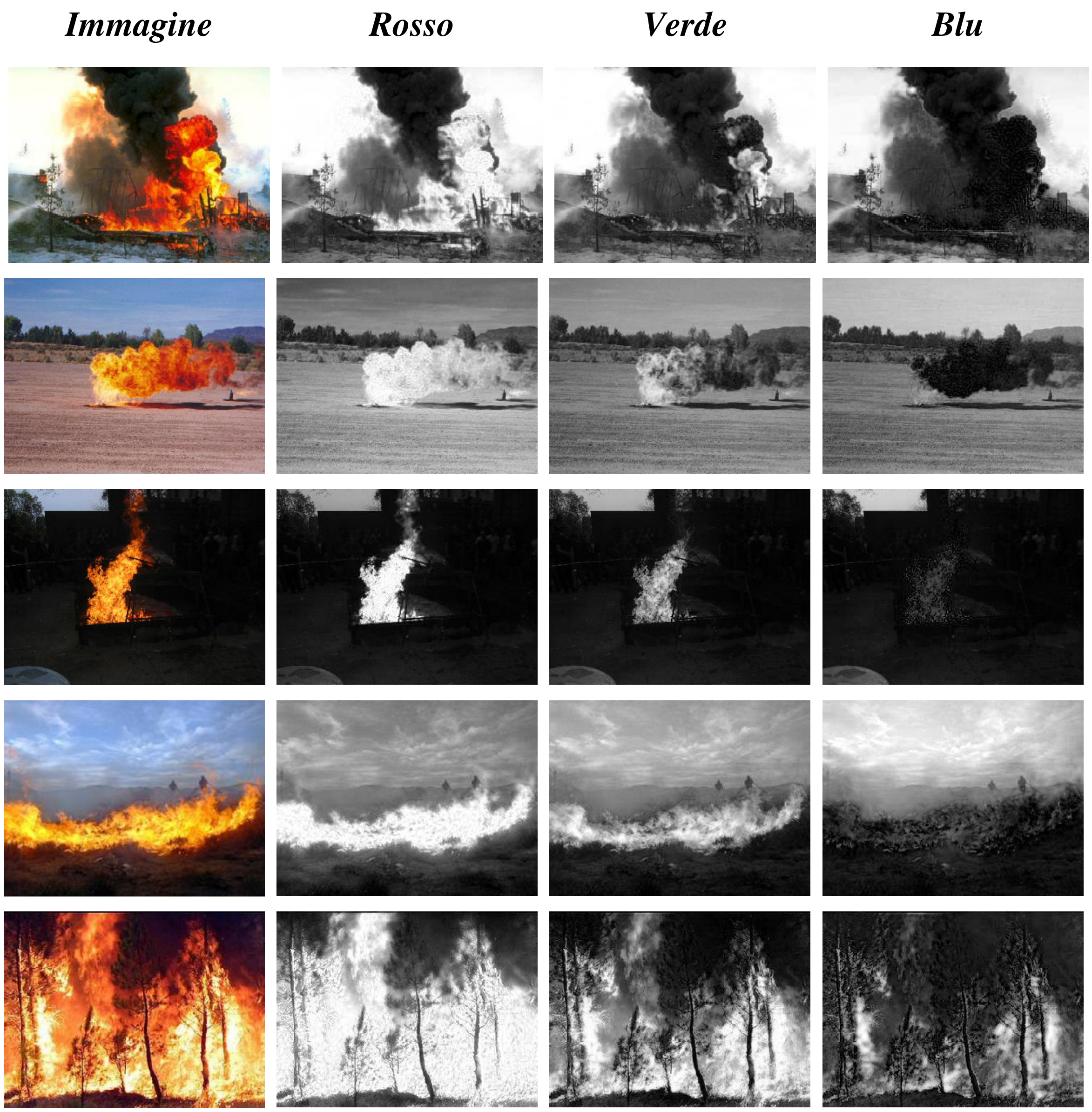


Tabella 1 - Nella tabella è possibile verificare su un campione di immagini limitato la validità delle restrizioni cromatiche imposte ai pixel

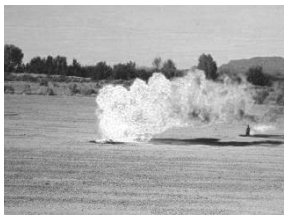
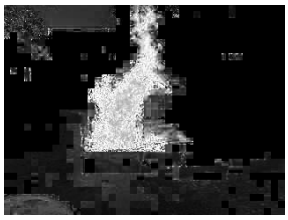
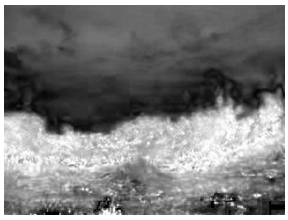
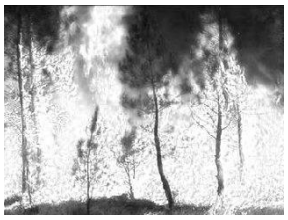
Immagine	*Saturazione*	*Luminosità*
		
		
		
		
		

Tabella 2 - Nella tabella è possibile verificare sul campione la validità delle restrizioni di saturazione e luminosità imposte ai pixel

Nelle aree interessate dal fuoco si è potuta notare una forte saturazione rispetto ai pixel del background, questo fatto è stato quindi tradotto nella

condizione che ogni pixel di fuoco debba avere una saturazione maggiore di 0.5, assumendo che la variabile S possa variare nel range [0…1].

Inoltre le aree interessate dall'incendio presentato alta luminosità, per cui si è imposto che un pixel di fuoco debba avere la componente V maggiore di 0.5 (sempre supponendo che questa variabile possa assumere valori nel range [0…1]).

Ognuno dei pixel non rispondente a questi criteri viene settato al colore nero (Red=Green=Blu=0), tali pixel non saranno sottoposti ad elaborazione.

Riepilogando, il filtro seleziona tutti quei pixel che soddisfano queste caratteristiche:

- Rosso > 128
- Blu < 128 $\vee$ Rosso < 200
- Saturazione > 0.5
- Luminosità > 0.5

5.3.3 Clusterizzazione dell'immagine

Una volta ottenuta l'immagine ottimizzata e filtrata di tutti quei pixel che sono sicuramente *non fuoco*, questa viene passata ad un algoritmo di clustering: nel caso specifico si è usato il K-Means. Nel sistema realizzato,

l'obiettivo è quello di suddividere l'immagine in un certo numero di gruppi di pixel ognuno caratterizzato da un certo criterio di somiglianza in modo da analizzare separatamente i cluster per la ricerca del potenziale incendio.

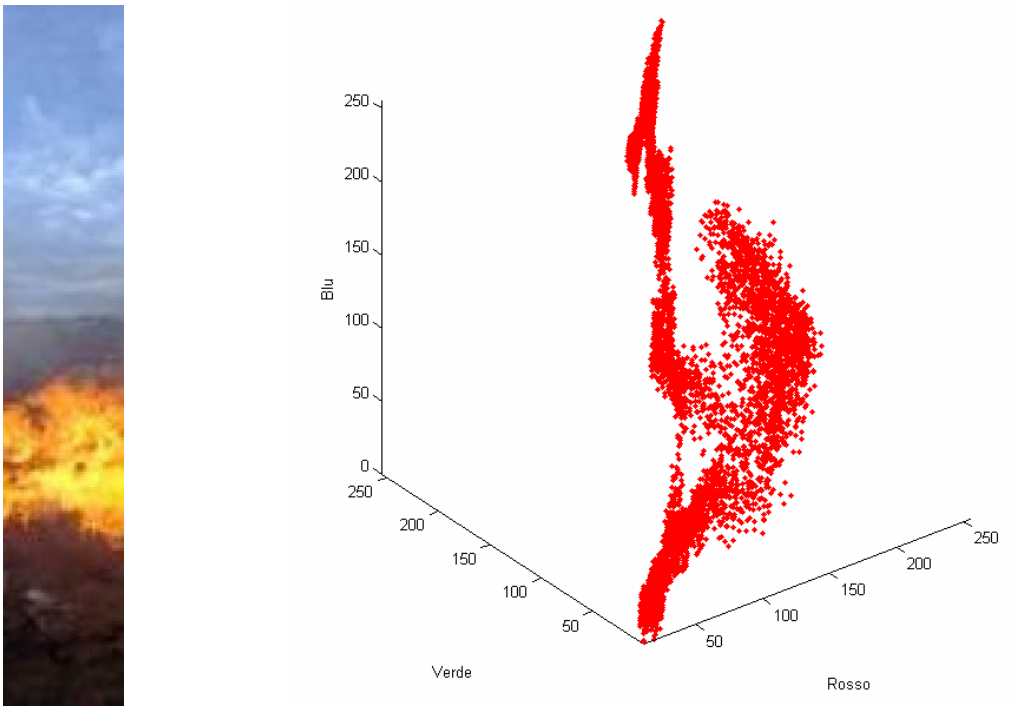


Figura 12 - Esempio di porzione di un'immagine e relativa proiezione nello spazio RGB

Nella figura (fig. 12) è possibile notare la proiezione dei pixel dell'immagine a sinistra, nello spazio RGB. La rappresentazione è data da una nuvola di punti, le cui coordinate (x,y,z) rappresentano le componenti RGB dei pixel.

A questo punto, un evidente ed immediato criterio di similarità applicabile ai dati in questione, è la distanza Euclidea, considerata nello spazio RGB, che

nell'immagine si tradurrà in una somiglianza cromatica dei punti in esame. Infatti pixel che hanno sfumature di colore simili si troveranno vicine nello spazio RGB, mentre punti con colorazioni diversi saranno lontani. Ad esempio, un pixel rosso ed uno arancione avranno una distanza inferiore rispetto ad un pixel rosso ed uno verde.

Per realizzare questo raggruppamento si è utilizzata un'implementazione esistente del K-Means. L'algoritmo funziona nel modo seguente:

$$[INDEX] = KMEANS\ (X, K)$$

- X è una matrice $N*P$, dove N è il numero di punti (pixel nel nostro caso) e P è la dimensionalità dei punti (in questo caso 3, R-G-B)
- K è un intero positivo che indica il numero di cluster che verranno creati
- *INDEX* è un vettore di N elementi contenente un intero compreso tra 1 e K. Indica l'indice a cui è associato l'n-esimo punto. In pratica, punti appartenenti ad uno stesso cluster avranno indice uguale

Nel caso trattato, si parte da una matrice X*Y*3 (X,U dimensione orizzontale e verticale in pixel dell'immagine, il 3 indica che ad ogni elemento di questa matrice sono associati 3 valori, ovvero quelli dello spazio RGB), e da essa si creano 3 vettori di dimensioni N=X*Y, che contengono i valori di R, G, B per ogni punto della matrice. Questi vettori vengono quindi inseriti in una matrice

di dimensioni N*3 che verrà usata come ingresso per l'algoritmo che implementa il *k-means* (parametro X). Per chiarire l'operazione effettuata dal *k-means* sull'immagine viene di seguito riportato un esempio della sua applicazione sulla figura 7 con un numero di cluster pari a 5:

$$indici = kmeans\ (\ X\ ,\ 5\ )$$

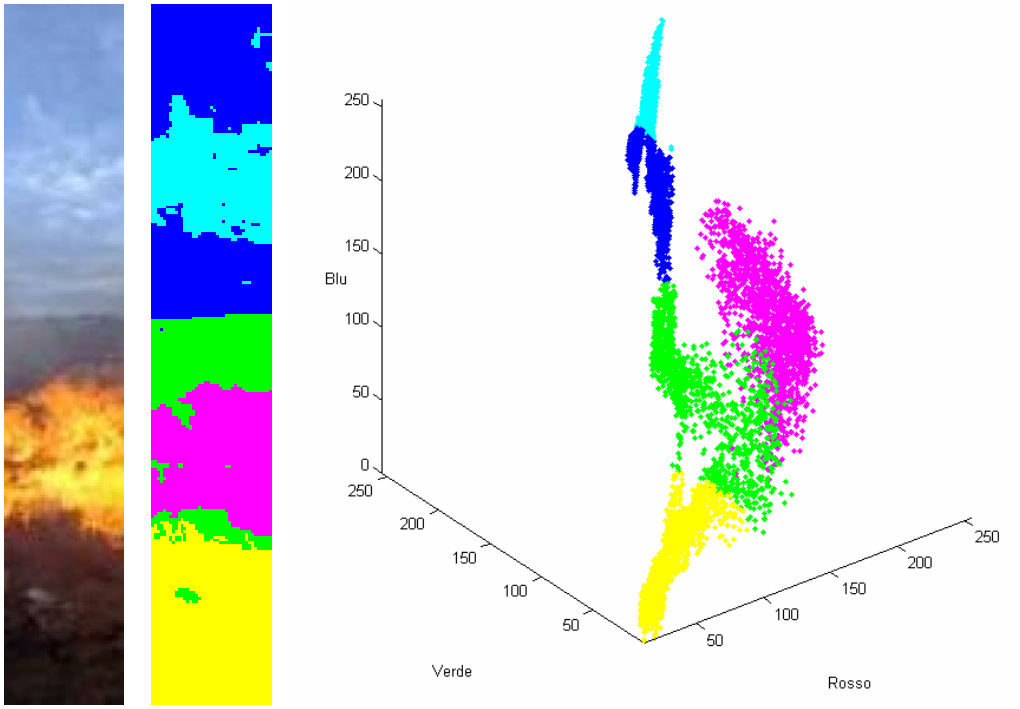


Figura 13 - Immagine originale, Immagine con evidenziati i cluster e disposizione nello spazio RGB dei cluster

A questo punto risulta evidente il meccanismo di funzionamento del clustering: l'immagine viene suddivisa in aree che hanno pixel simili per

colore; questo fatto viene appunto tradotto nello spazio RGB come è facile notare nel concetto di "vicinanza" geometrica.

Poiché l'algoritmo di clustering è molto oneroso dal punto di vista computazionale, nel sistema realizzato si è scelto di usare un numero di cluster pari a quattro (K=4). Nonostante un numero di cluster maggiore avrebbe garantito una maggiore distinzione delle aree di colore, ci si è dovuti limitare a soli 4 gruppi di colore, infatti ogni ulteriore cluster da calcolare fa aumentare il tempo di computazione in modo esponenziale. Questo fatto, risulta immediato se si pensa che oltre alle iterazioni nidificate del kmeans, nella fase successiva ognuno dei cluster subirà un'ulteriore elaborazione.

Inoltre, anche se il sistema è indipendente dalla risoluzione dell'immagine, non deve essere trascurato il fatto che le prestazioni del sistema diminuiscono all'aumentare del numero di pixel da elaborare (e quindi all'aumentare della risoluzione dell'immagine).

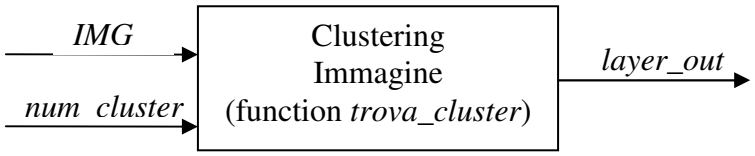
Sono state introdotte, inoltre, ulteriori ottimizzazioni all'algoritmo specificando altri parametri:

indici = kmeans(X, K, 'distribuzione iniziale uniforme', 'M iterazioni massime', 'azione cluster vuoto')

Oltre ai parametri già noti, viene ulteriormente specificato:

- *distribuzione iniziale uniforme*, distribuzione uniforme dei centroidi iniziali
- *M iterazioni massime*, numero di iterazioni massimo per trovare i *K* cluster
- *azione cluster vuoto*, quando un cluster perde tutti gli oggetti membri, ne viene creato uno nuovo con un punto che sia il più lontano possibile dal suo centroide

Il blocco di clusterizzazione opera nel seguente modo



Prende in ingresso *IMG*, l'immagine a colori in formato matriciale da clusterizzare (dimensioni Y,X,3), e *num_cluster*, un intero positivo indicante il numero di cluster da creare.

In uscita il blocco restituisce *layer_out*, una matrice tridimensionale di dimensioni Y*X*num_cluster, che contiene i cluster in cui è stata suddivisa l'immagine.

Un generico elemento di questa matrice, *x,y,i* potrà assumere i seguenti valori:

$$\begin{cases} 0, & \text{se il pixel x,y non appartiene all'i-esimo cluster} \\ 255, & \text{se il pixel x,y appartiene all'i-esimo cluster} \end{cases}$$

5.3.4 Elaborazione dei cluster

Una volta che l'immagine viene processata dall'algoritmo di clustering, saranno individuato quattro regioni, i cui pixel sono simili per colore. Queste regioni sono memorizzate in una matrice di tre dimensioni, due per la risoluzione in pixel dell'immagine ed un'altra che identifica un certo cluster. Nella seguente figura viene mostrata una rappresentazione grafica della matrice *layer_out*

Figura 14 - Un'immagine e la relativa rappresentazione della matrice layer_out

Ognuno di questi cluster verrà valutato separatamente dal blocco di classificazione dei pixel.

La scelta di utilizzare un algoritmo di clustering nelle immagini da analizzare è subentrata in un secondo momento rispetto allo sviluppo del sistema. Infatti, come verrà descritto in seguito, per stabilire se in un'immagine sia presente un incendio, si fa affidamento su una certa soglia, calcolata in base alla percentuale di pixel di fuoco rispetto al numero di pixel totali. Ovviamente questo parametro risulterà dipendente dalle dimensioni dell'incendio in relazione alle dimensioni dell'area monitorata dalla sorgente video. Infatti avere un incendio "piccolo" in un'immagine (a cui corrisponde una percentuale di pixel di fuoco bassa) può essere associato ad una grande distanza dell'incendio stesso dalla sorgente video.

Inoltre la percentuale di pixel di fuoco viene falsata da tutti i pixel e le aree di pixel segnati come fuoco ma che in realtà non lo sono.

Per evitare questi due problemi l'immagine è stata suddivisa in quattro cluster, i cui pixel sono in qualche misura somiglianti per colore. A questo punto interviene l'algoritmo di classificazione, che calcolerà la percentuale di pixel di fuoco relativa ad ogni cluster. In questo modo, l'eventuale incendio, anche se piccolo, andrà a ricadere in un cluster in cui la maggioranza dei pixel

appartengono all'incendio stesso questo proprio per le caratteristiche intrinseche di un incendio (alta saturazione, alta luminosità e contrasto, componente rossa elevata); la percentuale di pixel di fuoco relativa a questo cluster sarà quindi più alta rispetto a quella calcolata per l'intera immagine, in virtù del fatto che buona parte degli altri pixel ricadrà nei rimanenti cluster.

La vera e propria classificazione dei pixel avviene applicando la trasformazione del pixel stesso nel piano della crominanza, ovvero dalle componenti R,G e B si passa a quelle s,t: (R,G,B)→(s,t)

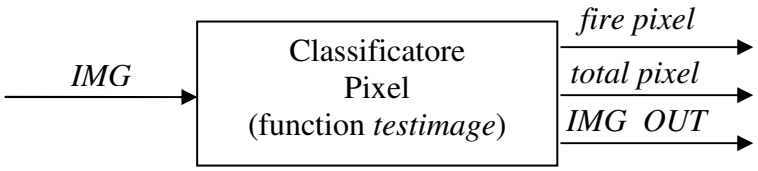
Successivamente tali componenti vengono scalate e traslate (secondo la notazione usata precedentemente per la creazione del modello) per essere conformi alla matrice del modello, le variabili s, t ricavate vengono usate come indice per controllare il valore relativo della matrice_modello. Se il valore relativo è nullo questo pixel viene segnato come *non fuoco*, mentre se il valore è strettamente maggiore di 0 il pixel viene segnato come *fuoco*. La segnatura del pixel viene effettuata settando a 255 una delle tre componenti R, G o B a 255 per caratterizzare i pixel di fuoco come intensi, mediamente intensi e poco intensi, secondo la seguente legge:

$$\begin{cases} 0 < \text{matrice\_modello} < 0.33 & \rightarrow R=0\ G=0\ B=255 \\ 0.33 < \text{matrice\_modello} < 0.66 & \rightarrow R=0\ G=255\ B=0 \\ 0.66 < \text{matrice\_modello} < 1 & \rightarrow R=255\ G=0\ B=0 \end{cases}$$

Ogni volta che un pixel di fuoco viene identificato la variabile *fire_pixel*, usata come contatore per i pixel di fuoco, viene incrementata di uno.

Il numero di pixel totali viene calcolato dalla risoluzione del cluster in ingresso (che sarà pari alla risoluzione dell'immagine), questo valore viene poi decrementato ogni qualvolta si trovi un pixel nero (R=G=B=0). I pixel neri identificano, infatti, quei pixel che sono stati esclusi dal cluster e che quindi non devono essere inclusi nel conteggio dei pixel totali.

Il blocco di classificazione si basa sul seguente schema,



in ingresso viene fornita un'immagine in formato matriciale, i cui elementi devono essere classificati secondo la proprietà *fuoco* o *non fuoco*.

Indipendentemente in questo blocco può essere passata sia un'intera immagine, sia un cluster trovato nel blocco precedente, ciò rende il blocco completamente slegato dal resto del sistema.

In uscita abbiamo tre variabili:

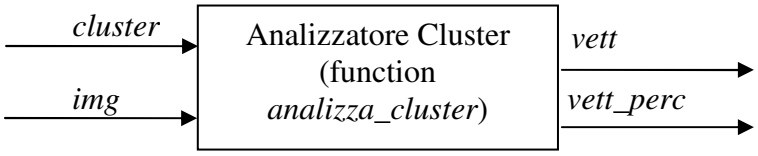
- *fire pixel*: indica il numero totale di pixel di fuoco rilevati nell'immagine (o nel cluster) specificati in ingresso
- *total pixel*: indica il numero di pixel totali presenti nell'immagine in ingresso. Tale parametro, nel caso in cui in ingresso si dia un'immagine, è banale: infatti in questo caso esso corrisponde alla risoluzione dell'immagine stessa. Nel caso in cui venga passato un cluster, devono essere esclusi dalla matrice data in ingresso tutti quegli elementi che indicano che un dato pixel non apparteneva al cluster
- *IMG_OUT*: rappresenta un copia dell'immagine passata in ingresso in cui però vengono evidenziati i pixel di fuoco, per avere quindi, una rappresentazione grafica della classificazione effettuata dall'algoritmo. I pixel di fuoco vengono rappresentati con 3 diversi colori Rosso Puro (R=255, G=0, B=0), Verde Puro (R=0, G=255, B=0), Blu Puro (R=0, G=0, B=255) (vedi fig. 15). Vengono usati tre diversi colori per indicare una differente probabilità che il pixel sia di fuoco.

Figura 15 - A sinistra l'immagine in ingresso al classificatore, a destra l'immagine in uscita in cui sono evidenziati i pixel di fuoco

5.4 Stima dei risultati e notifica degli allarmi

5.4.1 Risultati

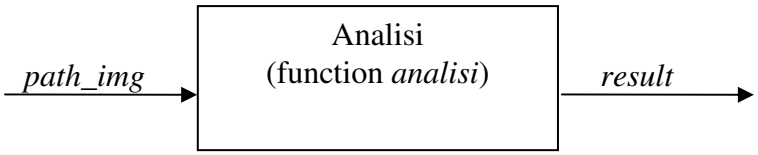
Per ogni cluster, il blocco classificatore ci fornirà due valori, il numero di pixel totali che ci permette di discriminare un eventuale cluster vuoto o molto piccolo, ed il numero di pixel che sono stati riconosciuti come 'fuoco'.



Il blocco che analizza tutti i cluster di una singola immagine (chiamato Analizzatore Cluster) prende in ingresso il numero di cluster da creare e la

matrice contenente l'immagini in formato RGB da analizzare. Tale blocco restituirà due vettori:

- il primo (vett), memorizza nel primo elemento l'indice del cluster con la percentuale di fuoco maggiore ed nei restanti elementi, le percentuali di pixel di fuoco contenute nei cluster segnati come contenitori di fuoco
- il secondo invece memorizza nell'elemento i-esimo la percentuale di pixel di fuoco del cluster i-esimo



Il blocco di analisi, invece supervisiona tutto il processo di fire detection producendo i risultati e memorizzandoli in un file di log. Per ogni immagine, specificata in ingresso dal suo percorso, il blocco avvia un timer e quindi l'elaborazione di tale immagine. Alla conclusione del processo di elaborazione il timer viene fermato ed il valore registrato (il tempo di esecuzione) viene memorizzato. Dal blocco precedente viene quindi ricavato:

- lo stato: 'Fire' o 'No Fire', questo viene ottenuto valutando semplicemente la lunghezza del primo vettore in uscita (vec_p), infatti

se la dimensione di tale vettore è maggiore di uno, è stato trovato un cluster che ha una percentuale di fuoco maggiore di quella della soglia

- la percentuale massima di fuoco riscontrata: l'elemento di indice uno nel primo vettore, contiene l'indice della percentuale di fuoco massima. Viene quindi ricavato tale valore dal secondo vettore (vett)

Queste due informazioni insieme alla durata dell'elaborazione vengono fornite in uscita da questo blocco di supervisione.

Infine, il blocco che fornisce in input una serie di immagini distinte da un numero sequenziale, si occupa di memorizzare nel file di log, l'indice dell'immagine elaborata ed il risultato del blocco di supervisione. I campi del risultato vengono separati da un punto e virgola ed i singoli risultati delle immagini vengono separati da un ritorno a capo. In questo modo il file prodotto è compatibile col formato CVS. Questo formato può quindi essere elaborato con degli opportuni strumenti di calcolo come Microsoft Excel o OpenOffice Calc.

Inoltre questi blocchi producono in uscita per ogni immagine in ingresso tre immagini che evidenziano le aree interessate dal fuoco.

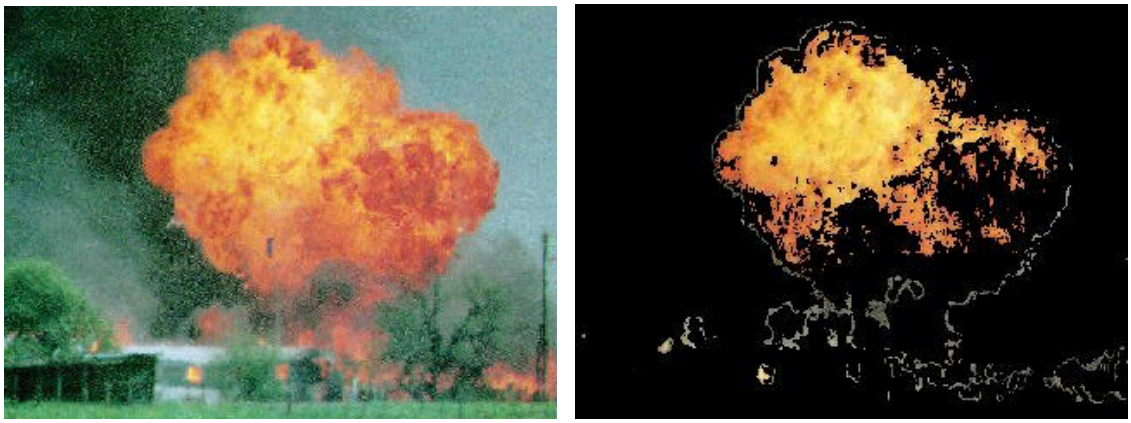


Figura 16 - Area interessata dal fuoco con colore originale

5.4.2 Invio dell'allarme ad utenti remoti, attivazione contromisure

Oltre agli allarmi ed alla normale attività del sistema che viene costantemente memorizzata in un file di log locale che è possibile consultare ed elaborare in maniera molto semplice con un programma di calcolo, ci si è posti il problema dell'invio degli allarmi ad un utente o ad un sistema remoto ed all'attivazione di ulteriori sistemi che permettano di gestire istantaneamente tramite intervento umano o automatico la potenziale situazione di pericolo. Per questo motivo quando un allarme critico viene rilevato, il sistema attiva una funzione che permette di eseguire un file con comandi per la shell del sistema operativo, a cui vengono passati alcuni parametri specifici, derivati dal tipo di allarme rilevato. Questa è stata la scelta ritenuta più efficiente in termini di velocità di elaborazione (non si

carica il sistema con l'esecuzione di ulteriori operazioni) e di facilità nell'estensione dei tipi di allarmi e sistemi di supporto attivabili: in questo modo l'utente non avrà la necessità di accedere al codice per adeguare il sistema alle proprie esigenze.

Per personalizzare le operazioni basta quindi modificare il file di shell (che nel caso specifico, in ambiente MS Windows è il file "alarm.bat") aggiungendo in esso i comandi opportuni per attivare le applicazioni che gestiranno la situazione di allarme.

Nel sistema presentato, viene mostrato un esempio di possibili allarmi inviabili:

- viene attivato un programma che invia una e-mail ad un certo indirizzo, specificando nel testo i parametri che hanno fatto passare il sistema in modalità allarme
- viene inviato un SMS ad un'utenza GSM con data e ora dell'invio

Ovviamente questo può essere esteso enormemente, per fare un esempio potrebbe essere inoltrata una chiamata verso alcuni utenti, potrebbe essere spedito un messaggio su una rete aziendale o direttamente su Internet a

determinati indirizzi, tutto in modo da rendere il sistema affidabile. Inoltre poiché la tempestività dell'intervento nei casi di incendio, soprattutto in zone a rischio, è un parametro fondamentale si ha quindi la necessità di attivare sistemi di contenimento automatici, in questo modo basterà inserire una riga che attivi l'operazione nel file suddetto.

5.5 Risultati Sperimentali

5.5.1 Risultati immagini contenenti fuoco

Nella seguente tabella viene schematizzato il risultato di test effettuati sul sistema. Nella colonna "% falsi positivi" viene indicata la percentuale di immagini contenenti fuoco ma che il sistema non riesce a riconoscere. Con modello generico si intende il modello creato con una serie di immagini contenente fuoco ma provenienti da diverse fonti; mentre con modello ad hoc si intende un modello creato con sole immagini provenienti dall'area monitorata.

	Numero immagini	Rilevamenti esatti	Falsi positivi
Modello generico	**67**	**92,2%**	**7,8%**
Modello ad-hoc	**70**	**95,4%**	**4,6%**

5.5.2 Risultati immagini non contenenti fuoco

Nella seguente tabella viene schematizzato il risultato di test effettuati sul sistema. Nella colonna "% falsi negativi" viene indicata la percentuale di immagini non contenenti fuoco ma che il sistema rileva come possibili incendi.

	Numero immagini	Rilevamenti esatti	Falsi positivi
Modello generico	**140**	**94,7%**	**5,3%**
Modello ad-hoc	**120**	**96,6%**	**3,4%**

CAPITOLO VI

Implementazione Software

6.1 Modello

Per la creazione del modello sono stati implementati gli script descritti di seguito, in particolar modo, il codice descritto nel paragrafo 6.1.3 è quello principale mentre tutti gli altri sono script contengono funzioni accessorie a tale scopo.

6.1.1 Limitazione Punti Modello (Limita.m)

Questo script permette di limitare un modello dell'incendio precaricato nello spazio di lavoro MatLab. I due cicli *for* annidati permettono di controllare tutti gli elementi della matrice modello. Se l'elemento (x,y) presenta un valore inferiore ad una certa soglia, quell'elemento verrà settato a zero, ovvero, il colore dei pixel associati a questo elemento verranno classificati come *non fuoco*.

```
soglia=0.13;
for x=1:length(modello(:,1))
     for y=1:length(modello(1,:))
          if (modello(x,y)<soglia) modello(x,y)=0;
          end
     end
end
```

6.1.2 Trasformazione RGB – Cromaticity Space (S_T_trasf.m)

La funzione s_t_trasf permette di passare dalle coordinate dello spazio RGB alle coordinate del piano della aromaticità (s,t). Vengono passate in ingresso le tre componenti R, G e B, in uscita vengono restituite le componenti s e t. Nel caso in cui la somma di R, G e B sia nulla (colore nero), e quindi si dovrebbe avere una divisione per zero, la funzione restituisce un valore di s e t pari a -999, valore che è fuori dal codominio della funzione di trasformazione. Questo valore indicherà appunto una situazione di errore.

```
function [s,t] = s_t_trsf(red,green,blue)
     rgbs = red+green+blue;
     if (rgbs==0) %controllo divisione per zero%
          s = -999; t=-999;
     else
          s = (2*red - green - blue) / rgbs;
          t = (green - blue) / rgbs;
          %trasformazione%
     end
```

6.1.3 Creazione Modello (Creagrafico.m)

La funzione crea_grafico ci permette di creare un modello a partire da un insieme di istantanee del luogo da monitorare, scattate in diverse condizioni

atmosferiche. Il modello creato sarà tanto più accurato quante più saranno le immagini processate e quanto più esse andranno a ricoprire le fasce orarie della giornata in diversi condizioni atmosferiche.

```
alfa=32;
beta=48;
num_immagini=15;
matrice_modello = zeros(3*alfa, 2*beta);
```

alfa e beta sono i parametri di scala, come descritto nel paragrafo relativo al piano (s,t), num_immagini indica il numero di immagini che dovranno essere incluse nel modello, la matrice matrice_modello viene inizializzata a zero e con dimensioni pari alla regione interessata dalla trasformazione (s,t), che sarà proporzionale ai fattori alfa e beta.

Viene quindi eseguito un ciclo *for*, per un numero di volte pari al numero di immagini specificato in precedenza. Ad ogni ciclo viene aperta una finestra di dialogo per indicare l'immagine da aprire. Questa viene quindi caricata in memoria come una matrice di pixel

```
%Ciclo acquisizione immagini del modello %
for x=1:num_immaggini
     %Caricamento immagine n° x %
```

```
[NomeFile, PercorsoFile] =
uigetfile({'*.jpg;*.tif;*.bmp', _
     'Image File';'*.jpg','JPEG File';'*.tif', _
     'TIFF File';'*.bmp','BitMap File';'*.*', _
     'All File'});
StrTmp = strcat(PercorsoFile,NomeFile);
I = imread(StrTmp);
clear StrTmp;
```

La funzione *uigetfile* permette di aprire file con una certa estensione restituendo il percorso del file ed il suo nome. Questi verranno usati come parametro della funzione *imread* che permette di aprire un'immagine come se questa fosse una matrice.

```
%Selezione del fuoco%
tmpIMG = roipoly(I);
redSel = immultiply(I(:,:,1),tmpIMG);
%livello rosso della selezione%
greenSel = immultiply(I(:,:,2),tmpIMG);
%livello verde della selezione%
blueSel = immultiply(I(:,:,3),tmpIMG);
%livello blu della selezione%
```

La funzione *roipoly* permette di selezionare tramite una spezzata poligonale una porzione di immagine, restituendo in uscita una matrice della stessa

dimensione della matrice di ingresso, fatti di uni e zeri. Lo zero indica che il pixel è all'esterno della regione selezionata, mentre l'uno indica che il pixel è interno. Moltiplicando quindi ogni livello di colore (rosso, verde, blu) per questa maschera (tmpImg) otterremo che l'immagine iniziale è nera nei punti non inclusi nella selezione, mentre manterrà il proprio colore nei pixel selezionati. Le matrice associate ad ogni livello di colore vengono quindi convertite in vettori grazie alla funzione *reshape*, e gli elementi in esse contenuti vengono trasformati da *unsigned int a 8 bit* a *double* per facilitare le operazioni successive.

```
%Conversione delle matrici RGB in vettori %
redSel=reshape(redSel,1,[ ]);
greenSel=reshape(greenSel,1,[ ]);
blueSel=reshape(blueSel,1,[ ]);
%Traformazione dei vettori dei componenti RGB in double%
redSel=double(redSel);
greenSel=double(greenSel);
blueSel=double(blueSel);
```

Ogni pixel verrà adesso trasformato secondo la legge (s,t), tale relazione viene applicata direttamente senza richiamare la funzione s_t_trasf, infatti questa opera su un singolo pixel, mentre nel caso seguente agiremo

contemporaneamente su tutti i pixel grazie alla capacità del MatLab di effettuare operazioni algebriche su matrici e vettori

```
%Trasformazione%
%NOTA: viene aggiunto + 0.0001 per evitare le divisioni per
zero %
s = (2*redSel - greenSel - blueSel) ./ (redSel + greenSel +
blueSel + 0.0001);
t = (greenSel - blueSel) ./ (redSel + greenSel + blueSel +
0.0001);
s=alfa*s;
t=beta*t;
s=round(s);
t=round(t);
```

Vengono arrotondati sia s che t proprio perchè vogliamo usare queste due variabili come indici della matrice modello, che non è altro che una rappresentazione discreta del piano s,t. Queste due variabili vengono anche traslate affinché gli siano conformi (positivi) agli indici della matrice modello.

```
%Traslazione%
s = s + (alfa + 1);
t = t + (beta + 1);
```

Vengono quindi ciclati i vettori s e t, ad ogni pixel i-esimo (vettori r, g e b) corrisponderanno gli elementi di indice i che avranno un certo valore di s e t. L'elemento della matrice corrispondente (s,t) verrà quindi incrementato di 1.

```
%aggiunta dati dell'immagine corrente al modello%
for x = 1:length(s)
     matrice_modello(s(x),t(x)) = matrice_modello(s(x),t(x))
     + 1;
end
```

Viene quindi mostrato il modello creato, successivamente al modello viene applicato un filtro mediano, grazie alla funzione *mdefilt2* che agisce su una matrice e viene richiamata con i suoi parametri di default, ovvero agirà su una sottomatrice 3X3. In seguito la matrice viene normalizzata rispetto al valore massimo ottenuto, in modo che il modello possa essere visualizzato come immagine in scala di grigi.

```
figure, imshow(matrice_modello)
matrice_modello_median = medfilt2(matrice_modello);
figure, imshow(matrice_modello_median)
matrice_modello_norm = matrice_modello_median ./
max(max(matrice_modello_median));
figure, imshow(matrice_modello_norm)
```

6.2 Analisi Immagini

6.2.1 Analisi sequenza immagini (Ciclo.m)

La funzione dà un'interfaccia da riga di comando, permettendo di analizzare un numero N di immagine caratterizzate da un certo prefisso e da una certa estensione, inoltre permette il logging dei dati prodotti dall'analisi delle immagini

In ingresso vengono passati 2 numeri che indicano il range delle immagini da analizzare, il prefisso di tali immagini e l'estensione, in pratica richiamando la funzione *ciclo(a,b,"img",".jpg")* saranno analizzate tutte le immagini *imgX.jpg* con X compreso tra a e b.

Viene prima richiamata la funzione *analisi* "catturato" il risultato di tale operazione e scritto sul file di log precedentemente aperto. La stringa col risultato è in formato CVS e quindi il file di log è gestibile da un qualunque programma di foglio di calcolo.

```
function ciclo(a,b,prefisso,suffisso)
fid = fopen('c:\log.txt','w');
for x = a:b
     fprintf ('************************ ITERAZIONE %d\n',x)
     img = strcat(prefisso,num2str(x),suffisso);
     result = analisi(img);
     result = strcat(num2str(x),';',result);
```

```
     fprintf(fid,result);
end
fclose(fid);
```

6.2.2 Analisi di un'immagine (Analisi.m)

La funzione analisi si occupa di coordinare le funzioni per la ricerca di fuoco in un'immagine. In ingresso viene passato il nome del file dell'immagine da analizzare ed in uscita viene restituita una stringa contenente il risultato dell'operazione.

```
function result = analisi(path_img)
stato=''; %inizializzazione della variabile stato (Fire , NoFire)
tic %Avvia un timer usato per il calcolo del tempo di elaborazione
img = imread(path_img); %carica in img l'immagine indicata in ingresso
layer = trova_cluster(img, 4); %cerca i cluster di un'immagine…
 [vec, vec_p] = analizza_cluster(layer, img, path_img); %...e vengono analizzati
duration = toc; %duration contiene il tempo trascorso tra tic e toc
img_tot = zeros(length(img(:,1,1)),length(img(1,:,1)),3); %inizializza la variabile img_tot alle dimensioni dell'immagine in ingresso
filename = strcat(path_img(1:(length(path_img)-4)),'a.jpg');
```

```
%viene create una copia del nome del file in ingresso con
aggiunto il suffisso a
if (length(vec)>1)
%vec è un vettore, che se ha dimensioni maggiori di uno
indica la presenza di %fuoco nell'immagine
     stato='Fire'; %stato viene settato a "Fuoco"
     percentuale=150; %percentuale ad un qualunque valore
     maggiore di %100
     for x=1:length(vec_p)
          if (vec_p(x)<percentuale)
               percentuale=vec_p(x);
          end
     end
     allarme(percentuale); %invio un allarme%
     %in questo caso vec_p contiene le percentuali di fuoco
     nei vari cluster, %in questo ciclo viene determinato
     quello con la percentuale più alta
     for x = 2:length(vec)
          img_tot = img_tot+layer(:,:,:,vec(x));
     end
     %vec contiene invece i cluster di pixel che sono stati
     individuati come %"Fuoco" vengono quindi sommati in
     unica immagine "img_tot"
     imwrite(img_tot,filename,'jpg');
     %che viene quindi salvata in un file
else
     %nel caso in cui non ci sia fuoco nell'immagine…
     percentuale=0; %setta a zero la percentuale di fuoco il
valore più basso
     for x=1:length(vec_p)
          if (vec_p(x)>percentuale)
```

```
                percentuale=vec_p(x);
            end
        end
        %cicliamo tutto vec_p alla ricerca del cluster con
        percentuale di fuoco %più bassa
        stato='NoFire'; %setto a "NoFire" lo stato
        fprintf '\nNo Fire!!\n'
        fprintf ('\nComunque il livello %d ha il massimo valore
        di fuoco!!\n',vec(1)); %visualizzo il cluster che aveva
        la percentuale di %fuoco più alta ma che non superava la
        soglia
end
result =
strcat(num2str(duration),';',stato,';',num2str(percentuale),'
\n');
%pongo result ad una concatenazione di stringhe, in esso sarà
indicato lo stato %dell'immagine, la durata dell'elaborazione
e la percentuale massima di fuoco %trovata
```

6.2.3 Ottimizzazione immagine (Ottimizza.m)

La funzione ottimizza.m prende in ingresso un'immagine e dà in uscita un'immagine che viene ottimizzata applicando opportune correzioni tramite l'uso di comandi disponibili in MatLab. In primo luogo viene effettuata una gamma correction sull'intera immagine, successivamente su ognuno dei piani R, G e B viene applicata un'equalizzazione dell'istogramma. Infine sui piani appena citati viene applicato un filtro mediano per eliminare eventuali rumori.

Dopo l'elaborazione, i tre piani vengono di nuovo fusi in un'unica matrice tridimensionale.

```
function imm=ottimizza(I)

J = imadjust(I,[],[],0.5);
%gamma correction%
J1 = histeq(J(:,:,1));
J2 = histeq(J(:,:,2));
J3 = histeq(J(:,:,3));
%equalizzazione dell'istogramma%
J1 = medfilt2(J1,[3 3]);
J2 = medfilt2(J2,[3 3]);
J3 = medfilt2(J3,[3 3]);
%rimozione rumore tramite filtro mediano%
J(:,:,1)=J1;
J(:,:,2)=J2;
J(:,:,3)=J3;
imm=J;
```

6.2.4 Filtraggio immagine (Filtro_Pixel.m)

La funzione Filtro_pixel ha il solo compito di prendere in ingresso un'immagine e filtrarla di tutti quei pixel che secondo analisi statistiche possono essere esclusi a priori dalla successiva elaborazione. La relazione secondo cui filtrare i pixel è la seguente:

Ovviamente tale filtro può essere migliorato in modo da escludere un maggior numero di pixel con l'effetto di una riduzione nei calcoli. Il filtro 'esclude' i pixel ponendoli pari al colore nero [R=255, G=255, B=255].

```
function imgout = filtro_pixel(I)
%dimensioni imm.%
dimy = length(I(:,1,1));
dimx = length(I(1,:,1));

H=rgb2hsv(I); %immagine nello spazio HSV%

for y = 1:dimy
    for x = 1:dimx
       if H(y,x,2) < 0.2
          I(y,x,1)=0;I(y,x,2)=0;I(y,x,3)=0;
       end
%condizione sulla saturazione
       if H(y,x,3) < 0.5
          I(y,x,1)=0;I(y,x,2)=0;I(y,x,3)=0;
       end
%condizione sulla luminosità
       if (I(y,x,3) > 128 & I(y,x,1)< 128)
          I(y,x,1)=0;I(y,x,2)=0;I(y,x,3)=0;
       end
       if I(y,x,1) < 128
          I(y,x,1)=0;I(y,x,2)=0;I(y,x,3)=0;
       end
     %condizione su Red e Blue%
```

```
    end
end
imgout = I;
```

6.2.5 Clustering immagine (Trova_Cluster.m)

La funzione trova_cluster divide l'immagine in un certo numero di gruppi specificati dal parametro d'ingresso num_cluster. I gruppi creati contengono pixel simili per colore, il parametro di somiglianza dall'algoritmo di clustering usato è la distanza euclidea calcolata nello spazio RGB. L'algoritmo di clustering usato è l'implementazione del k-means usata in Matlab. In uscita troveremo un vettore di immagini, ogni elemento del 'i' del vettore contiene un'immagine, che ha marcati in rosso i pixel appartenenti all'i-esimo cluster.

```
function layer_out = trova_cluster(img, num_cluster)
dimy = length(img(:,1,1)); %dimensioni immagine
dimx = length(img(1,:,1));
colore = zeros(num_cluster,3);
%matrice dei colori dei cluster
%livello rosso, verde e blu
red = img(:,:,1);
green = img(:,:,2);
blue = img(:,:,3);
%vettorizzazione dei livelli
red = red(:);
```

```
green=green(:);
blue=blue(:);
%dimensione vettori
dim = length(red);
%creo una matrice di colori dim * 3 (<-- R G B)
matr = zeros(dim,3);
matr(:,1)=red;
matr(:,2)=green;
matr(:,3)=blue;
%applico il kmeans del Matlab
Indici = kmeans(matr,num_cluster,'maxiter',60, _
'emptyaction','singleton');
%inizializzazione matrice in uscita dei cluster
layer_out = zeros(dimy,dimx,3,num_cluster);
%ciclo i pixel
for i = 1:dimy
    for j = 1:dimx
        %setto a rosso il pixel del cluster corrispondente
nella matrice dei cluster
        layer_out(i,j,1,indici((j-1)*240+i)) = 255;
    end
end
```

6.2.6 Analisi cluster (Analizza_Cluster.m)

Questa funzione ha il compito di prendere in ingresso un'immagine ed i gruppi di pixel rilevati dall'algoritmo di clustering. Ogni cluster dell'immagine viene analizzato separatamente. In uscita questa funzione creerà un numero di immagini scritte su file in formato JPEG pari al numero

di cluster. Tali immagini generate per scopi di debug contengono segnati in rosso tutti i pixel di fuoco rilevati nel cluster di appartenenza. Inoltre viene creato un vettore contenente le percentuali di fuoco rilevate in ogni cluster in modo da poter generare un eventuale allarme.

```
function [vett, vett_perc] = analizza_cluster(cluster, img,
path_img)
dimx = length(img(1,:,1)); %dimensioni immagine
dimy = length(img(:,1,1));
cc = 1; max = 0; maxl=0; %variabili temporanee
vett_perc(cc)=0; %inizializzazione
vett(cc)=0;
layer_fire=zeros(dimy,dimx,3,3); %matrice dei cluster
num_cluster=length(cluster(1,1,1,:));
for ll = 1:num_cluster %ciclo i cluster
    for x=1:dimx %e tutti i loro pixel
        for y=1:dimy
            if (cluster(y,x,1,ll)==255)
          %i cluster sono delle maschere dell'immagine. hanno
          colore rosso nel pixel che appartiente nero nel
          resto
                layer_fire(y,x,:,ll) = img(y,x,:);
              %copiamo il pixel appartente al cluster
              dall'immagine originale alla copia
            else
                layer_fire(y,x,:,ll) = [0 0 0];
              %se non appartiene al cluster settiamolo a
              zero (nero)
```

```
            end
        end
    end
[pxfuoco, pxtotali, imag]=testimage(layer_fire(:,:,:,ll));
%eseguo il test in modo da avere settati a rosso i pixel di
fuoco
    valore = pxfuoco / pxtotali * 100; %percentuale di pixel
di fuoco nel cluster ll
    vett_perc(ll)=valore;
%una percentuale del 30% è intensa come stato di allarme
quindi registro tutti i cluster con %pixel fuoco > 30 e
memorizzo anche quello che ha la percentuale massima
    if (valore>max)
        max=valore; maxl=ll;
        vett(1)=ll;
    end
    if (valore>30)
        cc=cc+1;
        vett(cc) = ll;
    end
%output utente
    fprintf ('Analisi cluster n° %d \n', ll);
    fprintf ('Ci sono %f percento pixel di fuoco \n',
valore);
end
[pxfuoco, pxtotali, imag] = testimage(img);
%eseguo il test su tutta l'immagine non clusterizzata in modo
da avere settati a rosso i pixel di fuoco, scrivo i risultati
in un file immagine%
filename = strcat(path_img(1:(length(path_img)-4)),'b.jpg');
imwrite(uint8(imag),filename,'jpg');
```

6.2.7 Analisi dei pixel (Test_Image.m)

Questa funzione è quella che effettivamente analizza ogni singolo pixel e lo marca come appartenente o non appartenente ad un potenziale incendio. In ingresso bisogna fornire un'immagine ed in uscita fornisce il numero di pixel totali (pari al numero totale di pixel dell'immagine esclusi quelli mascherati), il numero di pixel di fuoco rilevati (totalpixel e pixelfire rispettivamente). Inoltre l'oggetto AA è la copia dell'immagine in ingresso in cui vengono settati in rosso, verde o blu i pixel di fuoco, secondo il grado di somiglianza ad un pixel di fuoco. Questo marcatura avviene confrontando il pixel analizzato col modello del fuoco creato a priori con le funzioni apposite

```
function [pixelfire, totalpixel, AA] = testimage(img)
load('c:\matlab-lib\modellov2.1_rid.mat');
% carico il modello del fuoco
alfa=32;    %parametro di scala 1
beta=48;    %parametro di scala 2
pixelfire=0; %resettiamo i contatori dei pixel
pixelfire2=0;
AA = img; %copia di img in AA
A = double(AA); %converto a double
dimx = length(A(1,:,1));    %dimensioni immagine
dimy = length(A(:,1,1));
BB = zeros(dimy,dimx);
```

```
totalpixel=dimx * dimy; %setto totalpixel come tutti i pixel
dell'immagine
modello_median = medfilt2(modello);
%applico il filtro mediano al modello
px_esc=0;
for x=1:dimx %ciclo tutti i pixel dell'immagine%
    for y=1:dimy
     s = (2*A(y,x,1) - A(y,x,2) - A(y,x,3)) / (A(y,x,1) +
     A(y,x,2) + A(y,x,3) + 0.0001);
     %traformazione nello spazio (s,t)
        t = (A(y,x,2) - A(y,x,3)) / (A(y,x,1) + A(y,x,2) +
A(y,x,3) + 0.0001);
        s=alfa*s; %applico la scala
        t=alfa*t;
        s=round(s); %arrotondo tutto ad interi
        t=round(t);
        s = s + (alfa + 1); %traslazione
        t = t + (beta + 1);
        if (A(y,x)==[0 0 0])
     %se il pixel è nero decremento il contatore dei pixel
     totalpixel = totalpixel - 1;
        end
        if (((A(y,x,1)<30) & (A(y,x,2)<30) & (A(y,x,3)<30)) |
((A(y,x,1)>240) & (A(y,x,2)>240) & (A(y,x,3)>240)))
            px_esc=px_esc+1;
        else
        if (modello_median(s,t)>0) %stessa operazione per il
modello mediano %
            %copy_img_fire_area_median(y,x) = 255;
            pixelfire2 = pixelfire2 + 1; %aumento il
contatore dei pixel di fuoco%
```

```
            if (modello(s,t)<0.16)
                AA(y,x,1) = 0;
%setto a blu i pixel di fuoco dell'immagine in ingresso
                AA(y,x,2) = 0;
                AA(y,x,3) = 255;
                BB(y,x) = 255;
            end
            if ((modello(s,t)>0.15) && (modello(s,t)<0.21))
                AA(y,x,1) = 0;
%setto a verde i pixel di fuoco dell'immagine in ingresso%
                AA(y,x,2) = 255;
                BB(y,x) = 255;
                AA(y,x,3) = 0;
            end
            if (modello(s,t)>0.20)
                AA(y,x,1) = 255;
%setto a rosso i pixel di fuoco dell'immagine in ingresso%
                BB(y,x) = 255;
                AA(y,x,2) = 0;
                AA(y,x,3) = 0;
            end
        else
            %AA(y,x,:)=[0 0 0];
        end
        end
    end
end
BB = medfilt2(BB,[4 4]);
for y = 1:dimy
    for x = 1:dimx
        if (BB(y,x)>0)
```

```
            pixelfire=pixelfire+1;
        end
    end
end
fprintf('Pixel Esclusi:%d\n',px_esc);
```

6.2.8 Invio di allarmi (Invia_Allarme.m)

Questa funzione, in modo molto semplice sfrutta la capacità del MatLab di eseguire comandi esterni. In questo caso viene richiamato lo script di shell 'alarm.bat' passando come parametro la percentuale massima di fuoco rilevato. Lo script di shell a sua volta richiamerà tutti i comandi necessari all'invio dell'allarme in remoto o per l'attivazione delle contromisure.

```
function allarme(percentuale)
comando = strcat('alarm.bat ',percentuale);
%stringa del comando%
[status,result] = dos(comando,'-echo');
%ridirezione dell'output sulla shell del matlab,
memorizzazione stato di uscita del commando e del risultato%
if (result~=0)
     %se si verifica un errore ne visualizzo i dettagli%
    fprintf('Rilevato un errore nel processo di invio
allarme\n');
    fprintf('Risultato: ');
    fprintf('%s', result);
end
```

CAPITOLO VII

Conclusioni e sviluppi futuri

Il sistema si è rivelato abbastanza efficace dal punto di vista del riconoscimento di incendi avendo identificato almeno il 90% degli incendi nel caso di un modello generico, quindi ci si aspetta che, in un ambiente specifico tarando il sistema con un modello creato ad hoc, si riesca superare tale soglia. Inoltre anche il numero di falsi positivi è molto basso, ciò ci permette di dire che il sistema sviluppato sia molto affidabile.

D'altronde però, il sistema non risulta efficiente in termini di velocità risultando lento nel caso in cui si voglia elaborare i dati provenienti da una sorgente video (ad esempio una webcam). Infatti, anche ottimizzando il sistema, si riesce ad elaborare 'solo' un frame ogni 5 secondi. In alcuni ambiti questo può rappresentare un limite.

Questo problema è senza dubbio dovuto principalmente all'algoritmo di clustering che assorbe il 65% circa del tempo di elaborazione. In futuro si potrebbe sostituire l'implementazione MatLab del K-Means con una più efficiente.

Il restante tempo di elaborazione viene poi assorbito dall'algoritmo che confronta i cluster col modello di riferimento, per velocizzare questa fase basta semplicemente collocare il sistema su un calcolatore con hardware più potente in quante il confronto viene effettuato semplicemente usando dei cicli annidati.

Inoltre il sistema che nell'ambito di questa tesi è stato usato ed ottimizzato per la 'fire detection' può facilmente essere adattato per rilevare altre superfici con colori omogenei. Durante lo sviluppo del sistema, ad esempio, è stato creato un modello usando come campioni, zone che racchiudevano la pelle di alcune persone. Si è potuto quindi verificare che, sempre col solito grado di affidabilità, il sistema era in grado di riconoscere la pelle umana in foto che non erano state inserite nel modello.

Bibliografia

[1] - An Early Fire-Detection Method Based on Image Processing

Thou-Ho (Chao-Ho) Chen, Ping-Hsueh Wu, and Yung-Chuen Chiou

Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan 807. R.O.C.

[2] - Vision Based Fire Detection

Che-Bin Liu and Narendra Ahuja

Beckman Institute University of Illinois at Urbana-Champaign Urbana, IL 61801

[3] - Fire Detection In Tunnels Using An Image Processing Method

S. Noda, Japan Highway Public Corporation, Osaka, K. Ueda, Nagoya Electric Works Co., Ltd.,Aichi, JAPAN

[4] - A System for Real-Time Fire Detection

Glenn Healey, David Slater, Ted Lin, Ben Drda, A. Donald Goedeke Donmar Limited 901 Dover Drive Newport Beach, CA 92660

[5] - Conceptual Clustering of Text Clusters

Andreas Hotho, Gerd Stumme

Institute of Applied Informatics and Formal Description Methods AIFB, University of Karlsruhe, D–76128 Karlsruhe, Germany;
http://www.aifb.uni-karlsruhe.de/WBS, fhotho, stummeg@aifb.uni-karlsruhe.de

[6] - "U-Statistic Hierarchical Clustering"
R. D'andrade (1978): Psychometrika, 4:58-67

[7] - J. C. Dunn (1973): "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters",
Journal of Cybernetics 3: 32-57
J. C. Bezdek (1981): "Pattern Recognition with Fuzzy Objective Function Algoritms", Plenum Press, New York

[8] - J. B. MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability",
Berkeley, University of California Press, 1:281-297

[9] - Adobe RGB – www.adobe.com

[10] - Vehicle Detection in Color Images
Juan Carlos Rojas and Jill D. Crisman
Northeastern University, Boston, Massachussets, U.S.

Ringraziamenti

Questa è la parte più di difficile da scrivere di tutta la tesi, perché non essendo particolarmente bravo a scrivere credo che non riuscirò ad esprimere nel migliore dei modi la gratitudine che ho nei confronti di tutte quelle persone che mi hanno aiutato e mi sono state accanto durante questa bella avventura.

Ovviamente, il primo grazie va alla mia famiglia, che mi ha permesso e mi ha sostenuto per raggiungere quest'obiettivo. Spero che nonostante ci abbia messo troppo tempo sarete orgogliosi di me e spero di non deludere le vostre aspettative durante la specialistica.

Un sentito grazie all'ing. Spampinato per il supporto datomi nello sviluppo di questa tesi, fornendomi tutto l'aiuto e gli strumenti necessari (giuro che restituirò la webcam Logitech in cambio di Aibo).

Grazie a tutti gli amici del paese e d'infanzia: Vanessa (ma chi te lo ha fatto fare?!? ;D), Gianco (stupido sauro), Valeria, Saro B., Saro S. e Peppe (Zafferanesi acquisiti), Fabio (alias Iachino), Ture C.(testa di cuoio), Alfio (Go-Go), Roberto (Canna), Roberta (bella Rò), Frà(sciame!!), Anna, Angela, Vera ed Ing. Giuseppa che mi sono stati vicini e mi hanno incoraggiato nei momenti difficili.

Un grazie particolare ad una persona che sento vicina come un fratello, perché in tutte le occasioni mi è stato accanto sia come amico che come 'collega', chiarendomi, quando poteva, i miei dubbi 'ingegneristici', grazie Ture.

Grazie anche a tutto il gruppo di Catania Daniela, Domenico, Danielina, Mery, Kirby e gli altri, non so il motivo…ma ho sentito il dovere di nominarvi ;D.

Un doveroso grazie anche a Luca con cui ho studiato e preparato materie e progetti e con il quale ho trascorso ore in macchina da e verso Catania, con cui ho scambiato parecchi quaderni di appunti (tranquillo credo di non averne perso nemmeno uno!!).

Un grazie a quel gruppo di persone che ho conosciuto durante e dopo il tirocinio e con cui è nato un ottimo rapporto di amicizia (scusate la presunzione!). Camillo (puntuale, preciso e disponibile), Andrea (P., molto simpatico se solo lasciassi perdere WoW), Fabio (ottimo cuoco e giocatore), Daniele, il Maestro, Melo e Latino, che oltre ad essere colleghi in gamba sono anche incredibili compagni di risate.

Un particolare ringraziamento a Sandro, per la collaborazione nello sviluppo della prima parte di questa tesi, per avermi torturato chiedendomi come creare un istogramma di una foto, per avermi costretto a fare le immagini più mirabolanti per la sua tesi.

Un bau-grazie ad Aibo che inizialmente era il soggetto della tesi ma che ha poi intrapreso la vita da randagio e mi ha mollato, lasciandomi il solo ricordo di una variabile CVAIBO: mp_Ezio.

Un “ringraziamento” anche ad una persona senza la quale probabilmente mi sarei laureato un anno prima…

Sperando di non aver dimenticato nessuno concludo qui la mia tesi…

GRAZIE A TUTTI