Riconoscimento e classificazione automatica di cartelli stradali

Matteo Aldini, Filippo Berlini e Brando Mordenti, *Ingegneria e Scienze Informatiche LM, Università di Bologna*

***Sommario*—Lo scopo di questo progetto è quello di riconoscere e classificare la maggior parte dei cartelli stradali italiani a partire da una immagine di input. Lo sviluppo del progetto si compone di una serie di algoritmi in cui ogni output costituisce l'input per la fase successiva; si parte con la normalizzazione dell'immagine per accentuare le componenti RGB (color-space scelto come riferimento) dei pixel dell'immagine per poi effettuare la segmentazione del colore, passo necessario per separare oggetti d'interesse (i segnali) dallo sfondo. A questo punto si ha l'algoritmo di etichettatura delle componenti connesse, che consente di determinare quali sono le regioni in cui ci si aspetta un cartello. In queste regioni di interesse viene quindi effettuato il riconoscimento della forma tramite la trasformata di Hough e la conseguente classificazione confrontando il cartello in esame con le matrici di forma del training set.**

***Keywords*—VR, road signs, detection, classification, hough, LaTeX, paper.**

I. Requisiti del sistema

Il sistema che si intende sviluppare si inserisce nell'ambito della Smart Mobility e della Visione Artificiale e consiste in una serie di algoritmi per riconoscere automaticamente e classificare i cartelli stradali italiani a partire da un'immagine acquisita tramite una fotocamera. La segnaletica presa in esame è quella verticale: segnali di pericolo, segnali di indicazione, segnali di obbligo, segnali di precedenza e segnali di divieto. La pressoché totale maggioranza di questi cartelli ha forma circolare, triangolare o quadrata: in questo caso di studio l'attenzione verrà focalizzata su queste forme e su blu e rosso, ossia i due colori principali che compaiono nelle categorie di cartelli citate in precedenza. Il sistema parte dall'assunzione di base che la fotocamera produca come input all'algoritmo immagini di buona qualità e che i segnali non presentino caratteristiche di rotazione o distorsione troppo accentuate. Un'altra caratteristica fondamentale delle immagini di input è un'adeguata distanza tra il punto in cui viene scattata la foto e il cartello (di conseguenza quindi la grandezza del cartello nell'immagine): l'algoritmo dovrà riconoscere cartelli a distanza ragionevole dall'obiettivo; di conseguenza, cartelli troppo piccoli non vengono presi in considerazione. E' importante – inoltre - che l'algoritmo possa avere un ragionevole grado di indipendenza dalle condizioni atmosferiche e di illuminazione.

II. Sviluppo

Lo sviluppo della serie di algoritmi necessari per passare dall'immagine di input al segnale stradale riconosciuto é stato messo in atto prendendo come framework di riferimento VRLab.

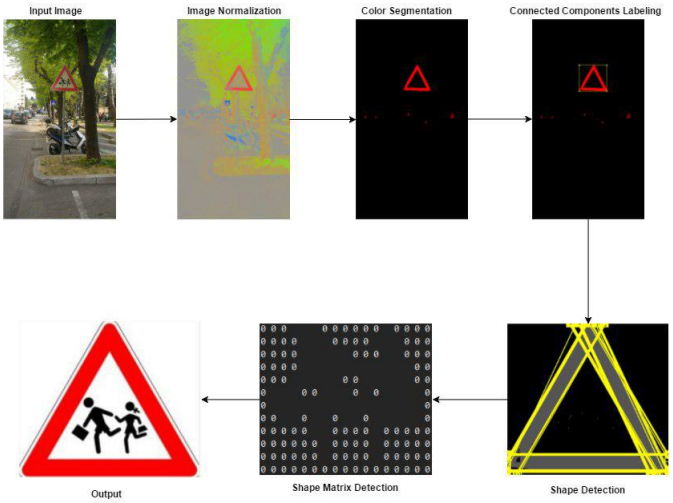

Figura 1. Schema di massima dell'algoritmo

$$sum = \sqrt{R_i^2 + G_i^2 + B_i^2}$$

$$R_o = \frac{R_i}{sum} \qquad G_o = \frac{G_i}{sum} \qquad B_o = \frac{B_i}{sum}$$

Figura 2. Formula utilizzata per la normalizzazione

A. Image Normalization

La prima operazione che viene effettuata sull'immagine acquisita è la normalizzazione nel color space RGB in modo tale da accentuare le componenti di red, green e blue all'interno dell'immagine. La procedura è composta da un semplice algoritmo che – data un'immagine *i* in input – la scorre pixel a pixel effettuando una trasformazione fino ad ottenere un'immagine normalizzata *o*. Tale trasformazione prevede (per ogni pixel) un valore *sum* dato dalla radice quadrata della somma dei quadrati delle tre componenti del pixel nell'immagine *i* (RGB). I valori delle tre componenti RGB del pixel corrispondente nell'immagine *o* saranno dati dal rapporto tra il corrispettivo valore della componente in *i* e il valore di *sum* moltiplicato per 255 (come illustrato in Figura 2).

Una volta ripetuto il procedimento per ogni pixel dell'immagine di input, ogni pixel dell'immagine di output avrà ottenuto dei nuovi valori normalizzati che semplificano e rendono più

Figura 3. Marcatura delle regioni sulle immagini

efficace la fase di segmentazione del colore.

B. Color Segmentation

La fase successiva alla normalizzazione è la segmentazione del colore, che permette di distinguere parti di interesse (in questo caso i segnali stradali) dallo sfondo e di binarizzare l'immagine, affidando ad un classificatore il compito di determinare (partendo da un training set) se l'oggetto in esame ricade al di sopra di una specifica soglia per poter essere considerato di uno specifico colore (in questo caso rosso o blu) oppure appartente allo sfondo. L'algoritmo si sviluppa in due parti principali: la fase di **training** e la fase di **classificazione**.

Nella fase di **training** viene istanziato un vettore di feature e vengono analizzate due immagini con regioni già *marked* in precedenza in corrispondenza delle parti blu e rosse dei cartelli stradali (è importante sottolineare che questo passaggio va effettuato con estrema precisione, per evitare che nelle immagini su cui si addestra il training set compaiano colori troppo distanti dalla media, troppo scuri o troppo chiari). Si scorrono quindi pixel a pixel le immagini marcate aggiungendo al training set i pixel interni alle regioni, sia per il rosso che per il blu.

A questo punto viene istanziato un **classificatore** di Bayes, due immagini di output (blu e rosso) e due matrici di interi che rappresenteranno delle label i cui indici rappresenteranno le coordinate dei pixel classificati come rossi o blu. Dopo di ciò si inizia a scorrere l'immagine estraendo le features per ogni pixel, classificandolo ed eventualmente colorando di blu/rosso il corrispondente pixel nella corrispettiva immagine di output. Oltre a cio, se il pixel viene classificato come blu o rosso viene anche messo a 1 il corrispondente valore nella matrice di label in modo tale che – una volta terminata di scorrere l'immagine – si possa risalire alle regioni di punti in cui sono presenti più pixel colorati rispetto alla media. Quest'ultimo passo rappresenterà l'input per la successiva fase di etichettamento delle componenti connesse.

III. Etichettamento delle componenti connesse

A partire dal risultato della segmentazione del colore, l'obiettivo è quello di identificare le regioni in cui compaiono un numero sufficientemente elevato di pixel adiacenti (1000 pixel). L'identificazione delle regioni avviene attraverso l'utilizzo del **connected components labeling**, in particolare attraverso il *two-step algorithm*. Ecco quali sono i due passi eseguiti dall'algoritmo:

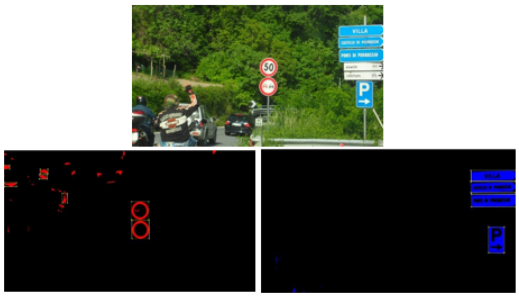

Figura 4. I risultati della segmentazione dei colori blu e rosso

A. Primo passo

- Si iterano tutti gli elementi dell'immagine per colonna e poi per riga
- Se l'elemento non è background (è stato classificato durante la color segmentation)
  - Si prendono i vicini dell'elemento corrente
  - Se non ha vicini, si effettua il labeling indipendente dell'elemento
  - Altrimenti, si effettua il labeling dell'elemento con la label più piccola tra quelle dei vicini
  - Si salva l'equivalenza tra le label vicine (attraverso una union-find data structure)

B. Secondo passo

- Si iterano tutti gli elementi dell'immagine per colonna e poi per riga
- Se l'elemento non è background
  - Si rieffettua il labeling dell'elemento con la più piccola label equivalente. In figura (X) lo pseudocodice che funge da linea guida per l'implementazione

Attraverso la struttura dati **Label** è possibile gestire l'equivalenza tra label. Ogni label ha una propria **Root**, che può essere aggiornata in passaggi successivi dell'algoritmo (il metodo **Join** permette l'aggiornamento delle Root in base alle equivalenze). Nel secondo passo dell'algoritmo, si considerano equivalenti le Label che posseggono la stessa Root.

Le regioni così individuate vengono salvate in delle strutture di tipo **RegionWithBounds**, in modo tale da ottenere i quattro corner (*upper-left*, *upper-right*, *bottom-left*, *bottom-right*) che racchiudono la regione. Le RegionWithBounds contenenti più di 1000 pixel saranno inserite nella **ImageWithMarkedRegions** relativa al colore per cui si è effettuata la segmentazione.

IV. Riconoscimento della forma

L'algoritmo di *shape detection* prende in ingresso le regioni restituite dal **connected component labeling** e decreta se all'interno della region è presente una delle forme di interesse, ovvero Triangolo Equilatero, Cerchio e Quadrato. Mentre la ricerca di Triangolo Equilatero e Quadrato si basa sull'analisi delle linee all'interno della regione attraverso l'**Hough Line**

```
algorithm TwoPass(data)
   linked = []
   labels = structure with dimensions of data, initialized with the value of Background

   First pass

   for row in data:
       for column in row:
           if data[row][column] is not Background

               neighbors = connected elements with the current element's value

               if neighbors is empty
                   linked[NextLabel] = set containing NextLabel
                   labels[row][column] = NextLabel
                   NextLabel += 1

               else

                   Find the smallest label

                   L = neighbors labels
                   labels[row][column] = min(L)
                   for label in L
                       linked[label] = union(linked[label], L)

   Second pass

   for row in data
       for column in row
           if data[row][column] is not Background
               labels[row][column] = find(labels[row][column])

   return labels
```

Figura 5. Two step algorithm

Detection, la ricerca del Cerchio si basa sull'analisi del centro più probabile attraverso la **Hough Circle Detection**.

A. Hough Line Detection

Vengono calcolate le 30 linee più probabili all'interno dell'immagine. Innanzitutto si calcola il modulo del gradiente dell'immagine, che viene successivamente binarizzato in base a un valore di *threshold*. A questo punto si scorrono tutti i pixel del modulo del gradiente binarizzato: ogni pixel, basandosi su coordinate polari, *vota* per le k rette che passano per quel punto, secondo intervallo dato dal parametro *thetaRange*. I risultati della votazione vengono salvati nella matrice A, i cui indici rappresentano i parametri theta e rho delle possibili rette dell'immagine. Le 30 rette più votate saranno quelle restituite sotto forma di **HoughLineParameters**. Un esempio di *line detection* si trova in Figura 6.

B. Hough Circle Detection

Il metodo utilizzato è analogo a quello descritto per il riconoscimento di linee, ma in questo caso i pixel basano il proprio voto sulla probabilità di essere il **centro** del cerchio che si sta cercando (le dimensioni del raggio vengono calcolate in base alla grandezza della regione). L'algoritmo restituisce per ogni pixel la un valore double che rappresenta la probabilità che tale punto sia il centro del cerchio che si sta cercando. In seguito si descrive l'approccio utilizzato per l'individuazione delle forme specifiche (Triangolo Equilatero, Quadrato e Cerchio).

C. Triangolo Equilatero

L'algoritmo si suddivide in due parti: **identificazione** dei potenziali triangoli equilateri all'interno della regione e **controllo sulla dimensione** dei triangoli identificati (la lunghezza del lato del triangolo deve essere all'incirca grande quanto la larghezza della regione d'interesse).

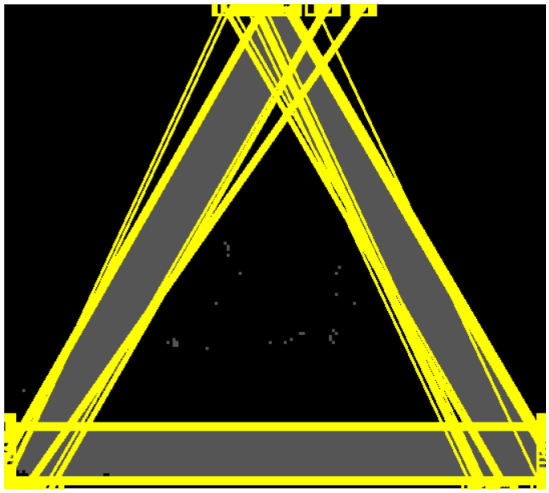

Figura 6. Hough Line Detction su un segnale di pericolo

- **Identificazione**: vengono controllate tutte le possibili combinazioni tra le rette precedentemente identificate. Tre rette identificano un potenziale triangolo equilatero. I triangoli equilateri sono quelli i cui angoli risultano all'incirca di 60. Il calcolo sugli angoli viene effettuato tramite il parametro *theta* delle rette.
- **Controllo sulle dimensioni**: la soglia di accettabilità per la grandezza del lato dei triangoli individuati è di *Width*-*Width*/5. Questo permette di scartare triangoli troppo piccoli che potrebbero essere stati identificati a causa del disturbo dell'immagine o erroneamente. Viene selezionato uno dei tre segmenti (un lato del triangolo) e se la sua lunghezza è maggiore della soglia si tratta di un triangolo equilatero accettabile.

È inoltre prevista la distinzione tra triangolo equilatero con base verso il basso (segnali di pericolo) o verso l'alto (dare la precedenza). La distinzione avviene controllando il parametro *rho* del lato con orientazione più vicina a 180.

D. Quadrato

L'algoritmo si suddivide in due parti: **identificazione** delle rette parallele e ortogonali e **controllo sulle dimensioni** dei possibili quadrati identificati da quelle rette (la lunghezza del lato del quadrato deve essere all'incirca grande quanto la larghezza o l'altezza della regione d'interesse).

- **Identificazione**: si controlla il parametro theta di tutte le rette precedentemente identificate. Le rette con *theta* simile sono ritenute parallele, le rette con *theta* che differiscono di circa 90 sono ritenute ortogonali tra loro.
- **Controllo sulle dimensioni**: si controllano le dimensioni del lato di ciascun possibile quadrato ottenibile con la combinazione di rette parallele o ortogonali. La soglia di accettabilità per la grandezza del lato dei quadrati individuati è di *Width*-*Width*/5 (se *Width* maggiore di

Height) oppure *Height* - *Height*/5 (se *Height* maggiore di *Width*).

E. Cerchio

L'immagine viene ridimensionata in modo da portare la larghezza pari a 100 pixel. Questo viene fatto in modo da poter interpretare in maniera uniforme il valore double restituito per ogni pixel dalla **Hough Circle Detection**. Prima di effettuare l'algoritmo di Circle Detection, si controlla che l'altezza e la larghezza dell'immagine siano simili, in quanto si sta ricercando un cerchio. Se il controllo viene soddisfatto, viene effettuata la Hough Circle Detection passando come parametro il **raggio** del cerchio da ricercare, ovvero *Width*/2 (con uno scarto di tolleranza). A questo punto si scorrono i valori dei pixel *centrali* restituiti dall'algoritmo (in quanto il centro del cerchio si troverà presumibilmente al centro dell'immagine) e si mantiene il valore più grande. Se questo valore supera la soglia di **0.06** l'immagine viene classificata come cerchio.

V. Classificazione basata su Shape Matrix

L'obiettivo della **shape matrix classification** è quello di effettuare la classificazione dell'immagine in input sulla base della sua forma e colorazione. Lo scopo è quello di segmentare la sagoma contenuta nell'immagine (la quale è un potenziale cartello stradale) e di confrontarla con le sagome estratte da tutti i cartelli del training set. Per il confronto tra le sagome viene utilizzata la tecnica denominata **Shape Matrix**:

- Si crea una griglia delle dimensioni dell'immagine con celle di dimensioni 3x3
- Si classificano i pixel dell'immagine in base al colore, che deve essere quello della sagoma che si sta analizzando
- Se in una cella della griglia compaiono pixel appartenenti alla sagoma, allora la cella viene riempita
- Il confronto tra l'immagine e l'immagine di training avviene sovrapponendo le rispettive *shape matrix*. La loro somiglianza sarà inversamente proporzionale al numero di errori di sovrapposizione

Le *shape matrix* delle immagini di training vengono precalcolate al momento dell'avvio dell'applicazione. Ecco i passi eseguiti dall'algoritmo di classificazione:

- Resizing dell'immagine di input: le immagini vengono rese delle stesse dimensioni di default
- Selezione dell'area di interesse: l'area in cui ricercare la sagoma interna al cartello varia in base alla forma del cartello stesso
- Estrazione della sagoma interna: attraverso segmentazione e binarizzazione, le tecniche utilizzate sono descritte in seguito
- Calcolo della *shape matrix* e confronto tra essa e le *shape matrix* di training
- La classificazione avviene in base al minor numero di errori di confronto

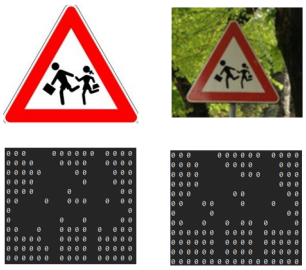

Figura 7. Confronto tra shape matrix di un cartello di test e della corrispondente immagine di training

A. Estrazione della forma interna

In base alla forma e al colore del cartello, le sagome da ricercare al suo interno hanno colorazioni differenti. Per esempio, in cartelli con forma quadrata o tonda e colore **blu**, la sagoma da ricercare all'interno sarà **bianca**, mentre in cartelli triangolari con bordo rosso la sagoma da ricercare sarà **nera**. In seguito si descrivono nel dettaglio le scelte effettuate per ciascun caso per l'estrazione della forma dall'immagine:

- Forma quadrata o tonda e colore blu: si effettua la segmentazione del colore **bianco**
- Forma triangolare e colore rosso: l'immagine viene binarizzata in base al minimo locale nell'istogramma dell'immagine (classe **SignBinarization**). La sagoma interna sarà data dai pixel **neri**. Questo metodo si è dimostrato più robusto della segmentazione del colore nero.
- Forma tonda e colore del bordo rosso: caso più complesso, in quanto la sagoma da ricercare può essere sia **nera** sia **rossa** (divieto di sosta e di fermata, divieto di accesso...)
  - Innanzitutto si effettua la **binarizzazione** dell'immagine per individuare un'eventuale sagoma nera. Mentre si effettua la binarizzazione, attraverso la segmentazione del colore rosso e blu si contano i pixel rossi oppure blu all'interno dell'immagine. La **segmentazione** è preceduta da **normalizzazione** dell'immagine. Se sono presenti più pixel rossi o blu del dovuto (più di 700) la sagoma da ricercare è rossa.
  - Se si deve ricercare una sagoma rossa, si effettua la **normalizzazione** dell'immagine e in seguito la **segmentazione** del colore rosso.

VI. Performance

In questa sezione si procede con la misurazione delle **performance** del sistema. In particolare, si analizzano le prestazioni in termini di numero di cartelli *classificati correttamente*, numero di cartelli *non classificati*, *errori* di classificazione e *falsi positivi*. Il test set utilizzato per questa fase è composto da immagini in alta definizione (HD) ottenute tramite la fotocamera di uno smartphone nelle zone di Cesena, Faenza e Imola con condizioni atmosferiche differenti e in diverse fasce

RISULTATI	
CLASSIFICATO	180
NON CLASSIFICATO	21
FALSO POSITIVO	5
ERRORE DI CLASSIFICAZIONE	11
TOTALE	217

Figura 8. Statistiche relative alla classificazione su un campione di 212 cartelli

CAUSE ERRORI	
Segmentation Error	16
Shape Detection Error	10
Shape Matrix Error	5
Connected Component Error	1
TOTALE	32

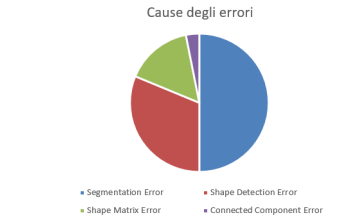

Figura 9. Statistiche relative alla natura degli errori di classificazione e delle mancate classificazioni

orarie (non notturne). Si effettuano le seguenti assunzioni di base:

- Cartelli **non sufficientemente grandi** poiché sullo sfondo non vengono presi in considerazione per il calcolo delle performance
- Cartelli **non presenti nel training set** non vengono presi in considerazione per il calcolo delle performance

Per ogni errore di classificazione o classificazione mancata viene specificata la *causa* dell'errore, ovvero l'algoritmo che l'ha causato.

Come si può notare, il sistema ha classificato correttamente 180 cartelli su 212, incorrendo in 5 falsi positivi. Gli errori di classificazione sono stati 11, mentre i cartelli non classificati 21. Per quanto riguarda i casi di mancata classificazione o di errori di classificazione le cause principali riguardano la segmentazione del colore e in minor parte la shape detection:

- La segmentazione del colore è cruciale nel nostro sistema, e un errore o imprecisione in questa fase si ripercuote in tutte le fasi successive portando ad errori. Date le differenti condizioni atmosferiche, le differenti angolazioni e le diverse fasce orarie in cui le foto sono scattate, è difficile raggiungere sempre una segmentazione del colore ottimale. In Figura 8 un esempio di errore dovuto all'erronea segmentazione del colore.
- Gli errori a livello di shape detection sono causati dalla tolleranza con cui l'algoritmo classifica determinate forme. Per evitare di incorrere in un numero elevato di falsi positivi, è necessario mantenere un livello di tolleranza non eccessivamente elevato sui vincoli che caratterizzano le forme (angoli del triangolo equilatero, angoli e dimensioni del quadrato, simmetria del cerchio). In figura è mostrato un errore dovuto al mancato riconoscimento del cartello nella regione individuata, causato dal fatto che l'angolazione della foto fa sì che non venga riconosciuto come triangolo equilatero.

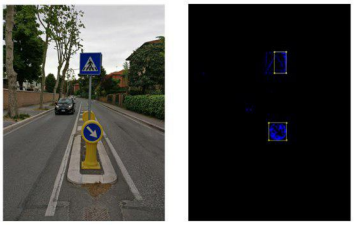

Figura 10. L'immagine risulta molto scura, per cui la segmentazione risulta imprecisa e causa il mancato riconoscimento del cartello di attraversamento pedonale

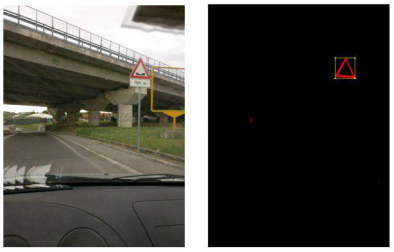

Figura 11. L'immagine di input è distorta, di conseguenza il triangolo appare come non equilatero e non viene riconosciuto

- In una minima percentuale dei casi, l'errore è stato causato dalla valutazione della shape matrix. L'errore in questo caso è causato dall'angolazione e dall'imprecisa segmentazione della forma, che porta ad un caso come quello mostrato in Figura 12 e 13. La segmentazione imprecisa, causata principalmente dalla separazione della forma dal resto del cartello, porta all'errata classificazione in quanto il minor numero di errori nel calcolo della shape matrix è riscontrato con il cartello di *pericolo di attraversamento animali*.

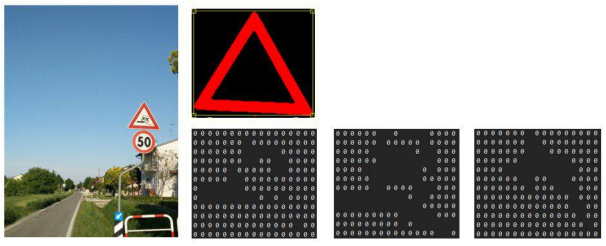

Figura 12. Errore di classificazione dovuto a inesattezze nella shape matrix

VII. Conclusioni

Le performance in fase di testing sono state ottenute dopo una accurata e iterativa parametrizzazione del sistema, con buoni risultati che dimostrano una discreta solidità anche in condizioni non ottimali. La maggioranza degli errori (il 50% del totale degli errori) sono stati determinati da errori di segmentazione del colore. In futuro si potrebbero valutare approcci differenti per il riconoscimento del colore, ad esempio dedicandosi esclusivamente allo studio dei canali RGB. La soluzione della Shape Matrix per la classificazione – nella sua semplicità - si è rivelata decisamente solida, generando solamente il 16% degli errori. Altre soluzioni percorribili potevano essere l'addestramento di classificatori o di reti neurali per la classificazione dei cartelli. Infine, uno sviluppo futuro di sicuro interesse potrebbe essere quello di rendere il sistema capace di riconoscere una maggiore varietà di cartelli.