

# Product Recognition on Store Shelves

Nicola Ferroni

## 1 Introduzione

Il progetto scelto si pone come obiettivo quello di sviluppare un sistema software in grado di eseguire l'analisi di immagini contenenti prodotti sugli scaffali di un supermercato.

In particolare si hanno a disposizione varie immagini di modelli di prodotti che devono essere riconosciuti, di ognuno di questi dovranno essere riportate la posizione all'interno della scena e le dimensioni del package.

Le scene su cui ci si propone di fare le analisi sono di tre diversi tipi, corrispondenti ai tre step di progetto proposti, ovvero: scene in cui un oggetto compare massimo una volta, scene in cui un oggetto può apparire più di una volta e scene a bassa risoluzione e con tanti oggetti da riconoscere, con eventuali occlusioni e prodotti non necessariamente identici a quelli presenti tra i modelli.

## 2 Primo Step

Nel primo step, come detto in precedenza, un oggetto può apparire una e una sola volta nella scena, che solitamente contiene un basso numero di oggetti ed è proposta ad una buona risoluzione.

Per questo tipo di problema è sufficiente l'applicazione delle Local Invariant Features, esattamente come visto in laboratorio, ovvero si trovano i keypoint e i descriptor all'interno di ogni modello e della scena, si esegue l'operazione di match (nel progetto è stato usato il FlannBasedMatcher, più efficiente rispetto all'approccio BruteForce) si filtrano i match secondo il test del rapporto tra distanze e si stima un omografia per ottenere i bounding box contenenti l'oggetto trovato nella scena.

### 3 Secondo e Terzo Step

Nel secondo step si aggiunge la possibilità che uno stesso package appaia più volte nella stessa scena, come suggerito nella descrizione del problema si è scelto di applicare ai keypoint trovati grazie al precedente step la trasformata di Hough generalizzata. Questa tecnica permette una precisione molto superiore rispetto al solo utilizzo delle Local Invariant Feature grazie al processo di voting.

Per farlo è quindi necessario trovare le corrispondenze tra i keypoint della scena e quelli del modello (facendo il contrario si troverebbero punti sparsi nella scena), poi nel modello si costruisce lo star model, ovvero si trova il baricentro dei KP e per ognuno di essi si trova il vettore che li unisce al baricentro, l'intensità del vettore verrà poi divisa per la scala indicata nel descrittore del KP.

Ora per ogni KP trovato nella scena andiamo ad applicargli il vettore ottenuto dal KP del modello con cui ha fatto match, moltiplicandolo per la sua scala, e si otterranno i cosiddetti voti (punti nello spazio).

Per l'analisi dei voti ho deciso di utilizzare un algoritmo di clustering basato sulla densità (Dbscan), questo permette di non dover campionare i voti secondo celle dello spazio di dimensioni difficilmente conosciute a priori, inoltre, anche conoscendo la dimensione ideale della cella, si ha sempre la possibilità che voti molto simili e vicini al centro finiscano in due celle diverse. Questo non sarebbe un problema se ci si pone come obiettivo quello di trovare i picchi, ma può diventare meno comodo se si vuole tenere traccia di tutti i voti che hanno avuto esito positivo, ciò sarà infatti utile nel momento in cui si dovranno tracciare bounding box più precisi possibile.

Analizzando i cluster ottenuti si potrà quindi capire quanti oggetti relativi ad un tal modello sono stati trovati nella scena e in quale posizione, interpreteremo inoltre il numero di elementi di un cluster come "peso", utile in fasi di confronto.

A questo punto sarà necessario trovare i bounding box di ogni oggetto trovato, rispetto al caso precedente questa risulterà più precisa visto che userò come KP solo quelli che hanno

prodotto un voto all'interno di un cluster.

In teoria il procedimento illustrato finora dovrebbe essere sufficiente per produrre risultati corretti, ma si è da subito notato che alcuni modelli sono molto simili tra loro e potrebbero essere riconosciuti entrambi nella stessa posizione.

Per risolvere questo inconveniente è stato scritto un algoritmo di filtraggio basato sul concetto di peso descritto poco sopra. In particolare, se il BB ha il baricentro contenuto dentro al BB di un altro oggetto si avrebbe una sovrapposizione impossibile per come vengono fisicamente organizzati gli scaffali, pertanto uno dei due deve essere scartato. Si sceglie quindi quello che ha collezionato meno voti.

L'algoritmo descritto risolve il problema delle sovrapposizioni, ma non sempre assegna il BB all'oggetto corretto.

Infatti i modelli di riferimento hanno dimensioni molto diverse, con conseguente aumento/perdita di dettaglio. Segue che un oggetto della scena relativo ad un modello poco dettagliato potrà essere riconosciuto da un altro modello simile (seppur diverso), ma più dettagliato.

Per diminuire questo problema si è scelto di procedere secondo questo criterio: se un oggetto viene trovato varie volte in una scena, tutte le occorrenze che hanno un peso inferiore ai 2/3 del peso massimo di quell'oggetto (si presume quindi che siano almeno leggermente diversi dagli altri che hanno avuto voti abbastanza più alti) verranno "penalizzate" e il loro voto verrà dimezzato, permettendo così al vero modello di superarlo.

Tuttavia neanche questo procedimento risolve completamente il problema: ci sono infatti package che hanno praticamente le stesse forme ma su sfondi di colore diverso, oppure package dello stesso prodotto ma appartenenti a serie o promozioni diverse, che hanno gli stessi elementi del modello sulla faccia principale, ma disposti in modo leggermente diverso rispetto al modello e più simili a ad altri modelli.

Si è deciso di implementare un confronto tra i colori dominanti di ogni package. Il riconoscimento dei colori dominanti viene

fatto tramite un altro algoritmo di clustering (K-means), che risulta però abbastanza pesante, soprattutto in immagini di grandi dimensioni. La soluzione adottata prevede quindi di applicarlo solo in caso di BB in conflitto e soltanto ad immagini drasticamente ridimensionate, infatti non è necessario avere immagini molto risolute per calcolare i 2 o 3 colori principali.

Tramite l'uso di quest'ultima tecnica si ottengono risultati sempre precisi per quanto riguarda le scene relative al secondo step.

Nelle scene del terzo step la funzione che si occupa di stimare l'omografia è spesso risultata poco robusta, soprattutto in caso di pochi voti andati a segno (seppur validi), dando forme molto diverse da rettangoli o lanciando eccezioni poco prevedibili. Si è quindi deciso di trovare il BB in un modo alternativo, più semplice dal punto di vista teorico, ma in questo caso più efficace.

Trovato un cluster, che per definizione dovrebbe approssimare il baricentro dei KP del modello nella scena, possiamo determinare il baricentro dei voti per avere una stima più precisa. A questo punto tracciamo nel modello un vettore dal baricentro dei KP al centro dell'immagine e, con un procedimento analogo al processo di voting della trasformata di Hough, applichiamo lo stesso vettore (opportunamente scalato) al baricentro dei voti nella scena, ottenendo così quello che sarà il centro del BB. Tramite i rapporti tra scale riusciamo quindi a calcolare i 4 punti del rettangolo.

Tutto ciò funziona solo con l'assunzione che gli oggetti siano correttamente posizionati sugli scaffali, ma, oltre a non avere mai mostrato comportamenti strani, riesce a trovare gli oggetti anche con pochissimi voti andati a segno.

Il funzionamento è quindi compatibile con tutti gli step proposti senza dover modificare nulla nel codice, di seguito sono riportati un esempio per ogni step.



Illustrazione 1: Esempio di riconoscimento in una scena di tipo 1



Illustrazione 2: Esempio di riconoscimento in una scena di tipo 2

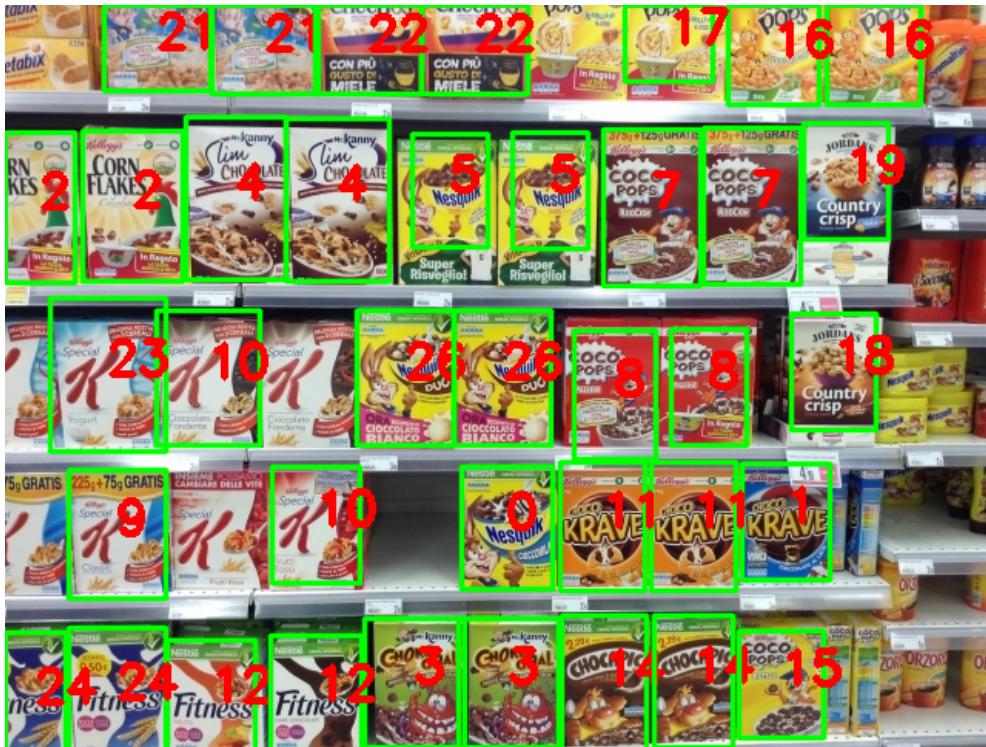


Illustrazione 3: Esempio di riconoscimento in una scena di tipo 3

## 4 Conclusioni

Come si può constatare osservando le immagini, il riconoscimento può essere considerato molto valido e preciso, mostrando incertezze solo nelle scene di terzo tipo e in caso di forti occlusioni o di incorrispondenza tra gli oggetti nella scena e i modelli.