Messina Mariagrazia

Progetto di Computer Vision:

Plugin per imageJ: Harry Corners Detection

Corso di laurea specialistica in Informatica.

A.A. 2006-2007

Introduzione

Il progetto realizzato consiste nell'implementazione dell'algoritmo di Harris e Stephens per la detection di corner in una immagine a toni di grigio. È stata introdotta una variante per tentare di rendere tale algoritmo invariante per scale, effettuando la ricerca dei corner in più immagini smussate e sottocampionate (piramide gaussiana).

Tale progetto è stato realizzato sotto forma di plugin per ImageJ (link).

Corners

I corners sono dei particolari tipi di features, cioè punti di particolare interesse in un'immagine, usati come punti di partenza in molti algoritmi di computer vision. È quindi fondamentale essere in grado di realizzare la detectin di tali elementi. Essa è un'operazione di image processing di basso livello, spesso è effettuata come prima operazione su di una immagine.

Un corner può essere definito come l'intersezione di due lati, un punto in cui ci sono edge con direzioni diverse. Esso deve avere una ben determinata posizione, in modo da rendere non troppo complessa la sua detection

Harris corners detection

Harris e Stephens migliorano il metodo di Moravec per la corner detection evitando di calcolare direttamente la funzione SSD come differenza tra porzioni di immagini vicine. I è l'immagine in input.

$$E(x,y) = \sum_{(x_k,y_k)\in W} w_k \left(I(x_k,y_k) - I(x_k+\Delta x, y_k+\Delta y)\right)^2$$

Per far ciò si considera lo sviluppo in seri si Taylor e tramite una serie di approssimazioni otteniamo

$$E(x,y) = (\Delta x \quad \Delta y) A(x,y) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

A è una matrice, strettamente legata alla funzione di autocorrelazione locale.

$$A(x,y)=\begin{bmatrix} \sum_{(x_k,y_k)\in W} w_k\left(I_x(x_k,y_k)\right)^2 & \sum_{(x_k,y_k)\in W} w_k I_x(x_k,y_k) I_y(x_k,y_k) \\ \sum_{(x_k,y_k)\in W} w_k I_x(x_k,y_k) I_y(x_k,y_k) & \sum_{(x_k,y_k)\in W} w_k\left(I_y(x_k,y_k)\right)^2 \end{bmatrix}$$

Dove la x e y a pedice della I rappresentano la tipica notazione delle derivate parziali. Per la realizzazione delle derivate è stato utilizzato il gradiente, come si fa abitualmente per il calcolo discretizzato della derivata prima, combinato con un filtro di smoothing (gaussiano), ottenendo in tal modo l'operatore Drog.

$$g(x,y,t)=\frac{1}{2\pi t}e^{-(x^2+y^2)/2t}$$

Dove t rappresenta la variaza.
La funzione peso w_k è una finestra gaussiana, la cui varianza è un parametro preso in input.
Gli autovalori α, β della matrice A sono proporzionali alle curvature principali dell'autocorrelazione locale e formano un descrizione invariante per rotazione della matrice A.
Tre casi possono essere considerati:

- se entrambe le curvature sono piccole (la funzione di autocorrelazione è piatta), la regione in esame ha intensità approssimativamente costante;
- se una curvatura è elevata e l'altra è bassa c'è un edge;
- se entrambe le curvature sono elevate c'è un corner.

La ricerca degli autovalori della matrice non è un'operazione computazionalmente leggera, ma non è necessaria in questo contesto, è sufficiente il calcolo del determinante e della traccia di A, per trovare i corners.
Ad ogni punto dell'immagine *I* viene associato un valore ***R*** (chiamato corner response) funzione degli autovalori α e β (viene così mantenuta l'invarianza rotazionale). *R* viene così definito:
R=Det(A)-kTr(A)2
Si ricorda che:
Det(A)=αβ
Tr(A)=α+β
k è una costante, generalmente fissata tra 0.04 e 0.06, nello specifico, nell'applicativo realizzato si ha k=0.06.
Solo per grandi valori di R si è in prossimità di un corner.
Si considerano quindi solo i punti di massimo locale (valutati su R), su essi si realizza una sogliatura (il cui valore è stabilito dall'utente) ed infine si effettua una selezione basata sulla distanza tra i corner trovati, in modo da

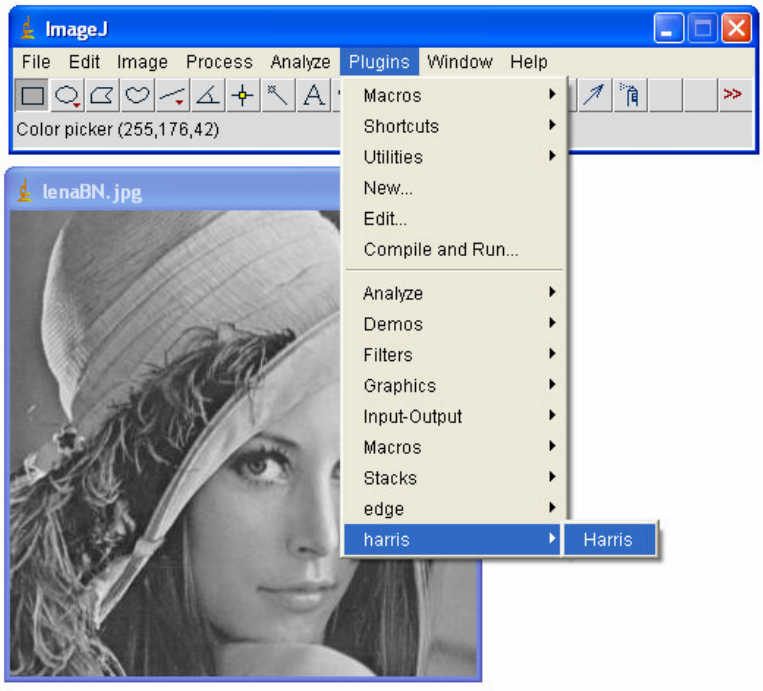
non ottenere molti features point nella stessa zona dell'immagine. Anche tale parametro è stabilito dall'utente.
Per avere una sorta di invarianza alla scale l'immagine in input viene smussata da un kernel gaussiano e sotto-campionata in modo da realizzare una rappresentazione scale-space . I corners vengono quindi ricercati nell'immagine originale (su cui verranno visualizzati)e in tutte le immagini sottocampionate, il cui numero è parametrizzato. Naturalmente prima di "disegnare" i corners è necessaria una mappatura delle coordinate dei punti nell'immagine con dimensioni originali.
Se il numero di sotto-campionamenti da effettuare scelto è troppo alto, la ricerca dei features point verrà effettuato senza l'utilizzo della piramide gaussiana.

Utilizzo del plug-in

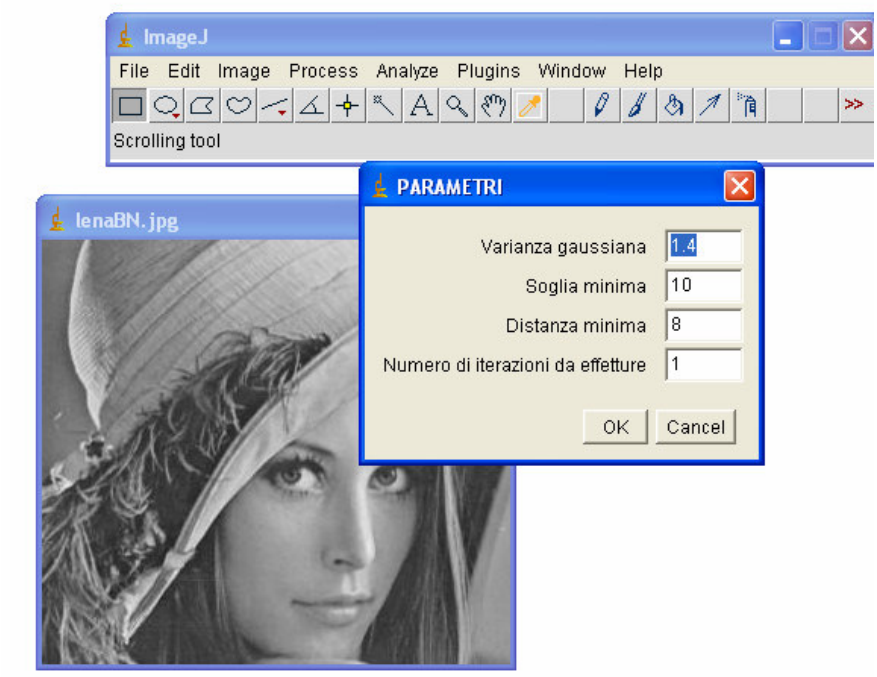
Per utilizzare il plugin realizzato è necessario innanzitutto scaricare ed installare ImageJ (link). A questo punto è possibile inserire nella cartella \ImageJ\plugin la cartella "harris" contenente i file class del progetto.

Dopo l'avvio di ImageJ, sarà necessario aprire un'immagine, selezionando la voce "apri" dal menù file. Si ricorda che il progetto realizzato funziona con immagini a toni di grigio, sarà quindi necessario aprire un'immagine di questo o tipo, o alternativamente convertirla con l'ausilio di imageJ, sfruttando l'opzione Type → 8 bit-gray dal menù image.

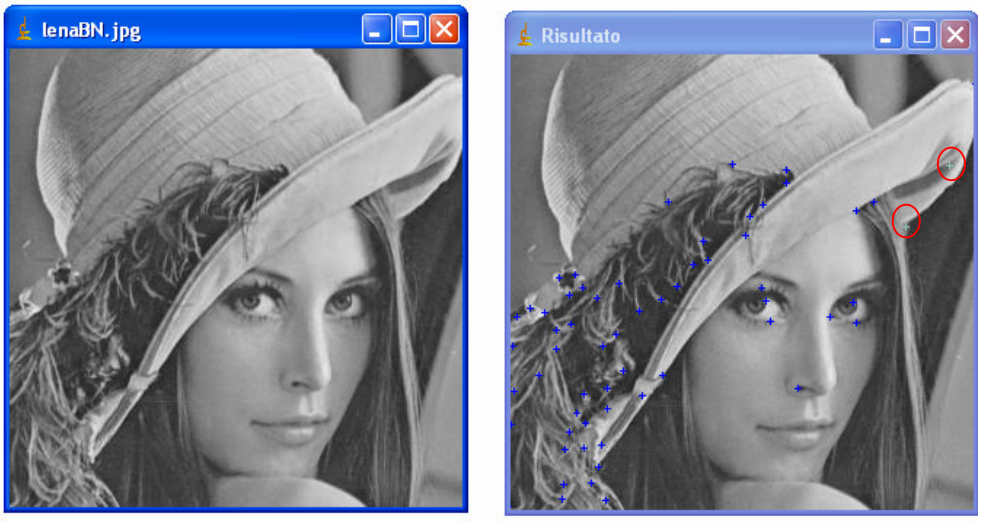
Nel menù "plugin" sarà presente la voce "Harris". Selezionando tale scelta, il plugin verrà avviato sull'immagine aperta in precedenza.



Per prima cosa viene chiesto all'utente di inserire i parametri che utilizza l'algoritmo per la ricerca dei corners. Sono comunque presenti dei valori di default, che possono essere semplicemente confermati.



In output si avrà una nuova immagine i cui corner saranno identificati come delle piccole croci. I colori diversi delle croci identificano i corners identificati nelle varie immagini presenti nella piramide gaussina, ottenute dal processo di sottocampionatura.



Progettazione

Sono stare realizzate 3 classi, quella principale, Harris_ che implementa l'interfaccia PlugInFilter e due di supporto, la classe GradientVector, che si occupa del calcolo della derivata discreta dell'immagine smussata e la classe supporto che contiene una serie di metodi sfruttati dalla classe principale.

DIAGRAMMA DELLE CLASSI: