Università degli studi di Catania
Corso di Laurea Specialistica in Informatica

MULTIMEDIA

2006/2007

Prof. S.Battiato

Relazione del progetto:

Estensione del software ImageJ con un plugin che simula un algoritmo di autofocus digitale

a cura di:

Nicola Ciraulo
gmcola@gmail.com

INDICE

INTRODUZIONE

Il plugin "GM Autofocus System" implementa un sistema di autofocus. In particolare data in input una sequenza di immagini che dovrebbe rappresentare l'insieme di immagini acquisite da una fotocamera digitale durante l'escursione focale dell'obiettivo, permette di trovate l'immagine che possiede la miglior qualità di mesa a fuoco simulando le escursioni focali che seguirebbe un obiettivo nella realtà.

L'algoritmo realizzato rientra nella famiglia degli algoritmi con **miglioramenti iterativi** e nella classe degli algoritmi **Hill-Climbing.** In particolare per mezzo di **funzione di valutazione** ad ogni fotogramma viene associato un coefficiente di messa a fuoco il quale sarà più elevato per immagini maggiormente messe a fuoco.
L'idea di base è quella di spostarci lungo la curva rappresentata dalla funzione di valutazione al fine di trovare il massimo globale che corrisponde all'immagine più nitida.
L'algoritmo funzionerebbe a meraviglia se non ci fossero massimi locali, infatti in tali punti un algoritmo banale potrebbe bloccarsi.
Per ovviare a questo problema sono state introdotte nell'algoritmo una serie di meccanismi che permettono di superare massimi locali ammesso che la valle che li separa da un massimo più grande non sia troppo profonda e/o troppo estesa.
In particolare se la valutazione del fotogramma corrente è migliore di quella del fotogramma precedente abbiamo a che fare con un passo in “salita”, ed in tal caso semplicemente continuiamo la ricerca.
Nel caso in cui si verificasse un passo in discesa la ricerca non si blocca subito restituendo il massimo trovato fino a quel momento ma permette un certo numero di passi “cattivi” in funzione delle oscillazioni della funzione di valutazione. Superato però un certo numero di passi in discesa la ricerca fa **backtracking**, cioè riparte dalla posizione dell'ultimo massimo trovato (o dalla posizione iniziale se abbiamo appena avviato la ricerca) e cerca nella direzione opposta. Se anche in questa maniera la situazione non migliora allora si ferma la computazione e si restituisce il massimo globale trovato.

La sigla **“GM”** presente nel nome del plugin è dovuta al fatto che consacro ogni mio lavoro al Signore Gesù Cristo e alla sua Santissima Madre la Vergine Maria.

LE CLASSI

Il plugin è costituito dalle seguenti classi:

- GM Autofocus
- ListenRoi
- SyncSliceStackIndex
- SyncSliceStackLabel
- InfoHelp

che verranno trattate detagliatamete di seguito.

Class GM_Autofocus

Questa è la classe principale del plugin, estende la classe PlugInFrame ed implementa l'interfaccia ActionListener.

La classe ha i seguenti metodi:

- *GM_Autofocus()*
  *C*ostruttore della della classe. Compie alcune semplici operazioni relative ai settaggi iniziali.

- *public void run(String arg)*
  Tale metodo si occupa dell'inizializzazione dell'interfaccia.

- *public void actionPerformed(ActionEvent e)*
  Tale metodo intercetta i click sui due pulsanti principali che sono "Load" e "Search". Analizzeremo dettagliatamente gli insiemi di istruzioni associati ai due pulsanti.

Pulsante "Load"

Il codice di tale pulsante si occupa della realizzazione dello stack "Start Point" che oltre a permette di vedere la qualità di messa a fuoco di tutte le immagini della sequenza in input permette di selezionare appunto la posizione iniziale da cui fare partire la ricerca. Segue una descrizione dettagliata delle istruzioni.

- Per prima cosa si setta il riferimento all'immagine attiva e si controlla che sia valido. Esso sarà poi utilizzato per il resto dell'algoritmo.
- Si controlla che l'immagine sia uno stack(una sequenza di immagini).
- Si controlla che l'immagine sia di tipo COLOR_RGB oppure di tipo GRAY8.
- Si controlla che sia stata effettuata una selezione, e che la stessa sia rettangolare. Se uno dei controlli precedentemente descritti fallisce si termina l'esecuzione del metodo e si torna nello stato di attesa.
- A questo punto per ogni immagine dello stack
  - si applica un filtraggio spaziale con i kernel che permettono di trovare i bordi nella sottoimmagine definita dalla selezione rettangolare.
  - Si effettua una stima della qualità di messa a fuoco sommando i valori dei pixel della sottoimmagine filtrata e dividendo tale valore per il numero totale di pixel.
  - Il valore ottenuto viene messo da parte in un array perchè sarà necessario per generare lo stack "Star Point". In particolare ogni immagine di tale stack è costituita dalla curva che rappresenta la valutazione di tutti i fotogrammi insieme ad un cerchietto che indica il punto in questione.
- Dopo si visualizza lo stack così ottenuto o semplicemente lo si aggiorna.
- Dopo di che si inizializzano le classi *SyncSliceStackIndex SyncSliceStackLabel* e *ListenerRoi.*
- Infine viene calcolato e visualizzato il tempo impiegato per il Load.

Pulsante "Search"

Il codice di tale pulsante si occupa della ricerca effettiva dell'immagine con la migliore messa a fuoco tra le immagini della sequenza in input. Alla fine dell'esecuzione verrà appunto visualizzata tale immagine. Segue una descrizione dettagliata delle istruzioni.

- Inizialmente si settano tutte le variabili necessarie all'elaborazione, tra le più importanti abbiamo:
  - *MAXINCREMENTO*
    che rappresenta il massimo numero di posizioni che si possono saltare in un passo.
  - *MAXSTEP*
    che rappresenta il numero totale di immagini della sequenza in input

- *MAXBADSTEP*

  che rappresenta il massimo numero di passi "cattivi" che si possono fare prima di cambiare direzione o di fermare la ricerca

- Viene avviato il ciclo che si occuperà della ricerca del massimo globale.
  - Al fotogramma corrente si applica un filtraggio spaziale con i kernel che permettono di trovare i bordi nella sottoimmagine definita dalla selezione rettangolare.
  - Si effettua una stima della qualità di messa a fuoco sommando i valori dei pixel della sottoimmagine filtrata e dividendo tale valore per il numero totale di pixel della selezione.
  - Il valore ottenuto viene messo da parte in un array perchè sarà necessario per generare lo stack "Steps". In particolare ogni immagine di tale stack è costituita dalla curva che rappresenta la valutazione di tutti i fotogrammi insieme ad un insieme di cerchietti che indicano tutti i passi effettuati fino a quel momento.
  - Si calcola la derivata della curva considerando la valutazione ottenuta nel punto precedente e quella nel punto corrente.

    Si calcola l'arcotangende della derivata per ottenere la pendenza della curva. Tale valore sarà poi utilizzato nel calcolo del numero di fotogrammi da "saltare". Si utilizza questo sistema perchè se la pendenza della curva è lieve allora assegniamo un numero elevato di fotogrammi da saltare, se la pendenza è elevata allora assegniamo un numero piccolo di fotogrammi da saltare, anche uno solo. Questo perchè si è visto sperimentalmente che in prossimità dei massimi la curva che rappresenta la valutazione della messa a fuoco assume una pendenza molto elevata.
  - Si tiene conto del numero di oscillazioni che la funzione compie. Infatti il numero massimo di passi "cattivi" è proporzionale al numero di oscillazioni in maniera tale da poter superare nella ricerca eventuali massimi locali a vantaggio di quelli globali.
  - A questo punto bisogna definire il fotogramma che verrà analizzato al passo successivo e compiere una serie di operazioni basilari per il sistema. Le operazioni da eseguire variano in base alle condizioni verificate, in particolare:

- *Se ci troviamo sul primo sull'ultimo fotogramma*

  - azzeriamo il contatore dei passi cattivi

  - cambiamo direzione di ricerca

  - continuiamo a cercare al di là della posizione iniziale saltando nella posizione indicata, perchè tanto siamo siamo sicuri che dalla posizione iniziale fino al primo o all'ultimo fotogramma abbiamo già cercato e non avrebbe senso rivalutare tali fotogrammi

- *Se facciamo un passo in salita*

  (valutazione attuale >= valutazione precedente)

  - azzeriamo il numero totale di passi cattivi

  - incrementiamo di 1 il contatore di passi buoni

  - settiamo il puntatore alla posizione precedente ed il puntatore alla posizione massima alla posizione corrente

  - settiamo la valutazione della posizione precedente ed la valutazione della posizione massima alla valutazione corrente

  - incrementiamo/decrementiamo il puntatore alla posizione successiva in base alla direzione di ricerca

  e continuiamo

- *Se facciamo un passo in discesa*

  (valutazione attuale < valutazione precedente)

  - *Se abbiamo raggiunto il numero massimo di passi cattivi, il numero di passi buoni è insufficiente e non abbiamo cambiato direzione fino a quel momento*

    - settiamo a vero il flag che indica il cambio direzione

    - azzeriamo il numero totale di passi cattivi

    - cambiamo direzione di ricerca

    - continuiamo a cercare al di là della posizione iniziale saltando nella posizione indicata, perchè tanto siamo siamo sicuri che dalla posizione iniziale fino a quella corrente abbiamo già cercato e non avrebbe senso rivalutare tali fotogrammi

  - *Se abbiamo raggiunto il numero massimo di passi cattivi con un numero sufficiente di passi buoni*

    - fermiamo la ricerca e ritorniamo alla posizione del massimo globale trovato fino a quel momento

        - *Altrimenti*

          - incrementiamo di 1 il contatore dei passi cattivi
          - continuiamo la ricerca in discesa saltando al fotogramma indicato e settando opportunamente i puntatori alla posizione corrente e successiva
    - Incrementiamo di 1 il contatore dei passi e se abbiamo superato il numero massimo di passi consentiti che è pari a 2 volte il numero dei totale dei fotogrammi fermiamo la ricerca e ritorniamo alla posizione del massimo globale trovato fino a quel momento
    - Visualizziamo nello stack in input il fotogramma corrente
- Settiamo il fotogramma visualizzato nello stack "StartPoint" col valore del massimo globale ottenuto in modo da predisporlo per una successiva ricerca
- Visualizziamo lo stack "Steps"
- Calcoliamo e stampiamo il tempo impiegato nella ricerca

Class ListenRoi

Questa classe si occupa dell'intercettazione dei cambiamenti nella selezione rettangolare ROI che rappresenta l'area di valutazione della messa a fuoco.
Per fare questo estende la classe "MouseAdapter" che permette di intercettare il momento in cui il mouse esce dallo stack delle sequenza delle immagini su cui effettuare la ricerca, il tale momento viene verificato che la ROI corrente sia identica a quella precedente, in caso affermativo non fa nulla, altrimenti disabilita il pulsante "Search" in modo da obbligare l'utente ad effettuare un nuovo "Load" prima di avviare al ricerca.

La classe ha i seguenti metodi:

- *ListenerRoi(ImagePlus i, GM_Autofocus gm)*
  *C*ostruttore della della classe. Si passa l'immagine in cui intercettare gli eventi del mouse ed un riferimento alla classe generale per mezzo del quale si potrà disabilitare il tasto "Serach".

- *public void mouseExited(MouseEvent e)*
  Ad ogni invocazione controlla che la ROI corrente sia identica a quella precedente, in caso affermativo non fa nulla, altrimenti disabilita il pulsante "Search" in modo da obbligare l'utente ad effettuare un nuovo "Load" prima di avviare al ricerca.

Class SyncSliceStackIndex

Questa classe opera la sincronizzazione tra la slice visualizzata nello stack "Start Point" e la slice corrispondente nello stack della sequenza delle immagini in cui effettuare la ricerca.
Per fare questo implementa l'interfaccia "ImageListener" che permette di intercettare i "cambiamenti" nello stack "Start Point" e di conseguenza effettuare l'aggiornamento.

La classe ha i seguenti metodi:

- *SyncSliceStackIndex(ImagePlus s, ImagePlus d)*
  *C*ostruttore della della classe, le due immagini passate in input rappresentano la sorgente da intercettare e la destinazione da aggiornare.

- *public void imageClosed(ImagePlus imp)*
  Non fa nulla.

- *public void imageOpened(ImagePlus imp)*
  Non fa nulla.

- *public void imageUpdated(ImagePlus imp)*
  Ad ogni invocazione setta nell'immagine di destinazione la slice che la lo stesso indice della slice visualizzata nell'immagine sorgente.

Class SyncSliceStackLabel

Questa classe opera la sincronizzazione tra l'etichetta della slice visualizzata nello stack “Steps” e la slice corrispondente nello stack della sequenza delle immagini in cui è stata effettuata ricerca.
Per fare questo implementa l'interfaccia “ImageListener” che permette di intercettare i “cambiamenti” nello stack “Steps” e di conseguenza effettuare l'aggiornamento.

La classe ha i seguenti metodi:

- *SyncSliceStackLabel(ImagePlus s, ImagePlus d)*
  *C*ostruttore della della classe, le due immagini passate in input rappresentano la sorgente da intercettare e la destinazione da aggiornare.

- *public void imageClosed(ImagePlus imp)*
  Non fa nulla.

- *public void imageOpened(ImagePlus imp)*
  Non fa nulla.

- *public void imageUpdated(ImagePlus imp)*
  Ad ogni invocazione setta nell'immagine di destinazione la slice che la lo stesso indice dell'etichetta della slice visualizzata nell'immagine sorgente.

Class InfoHelp

Questa classe ha il compito di visualizzare una finestra di testo che contiene una breve descrizione del plugin. Implementa l'interfaccia ActionListener per intercettare i click sul pulsante “Info & Help”.

La classe ha il seguente metodo:

- *actionPerformed(ActionEvent e)*
  Ad ogni invocazione apre un finestra di testo che contiene una breve descrizione del plugin.

INTERFACCIA

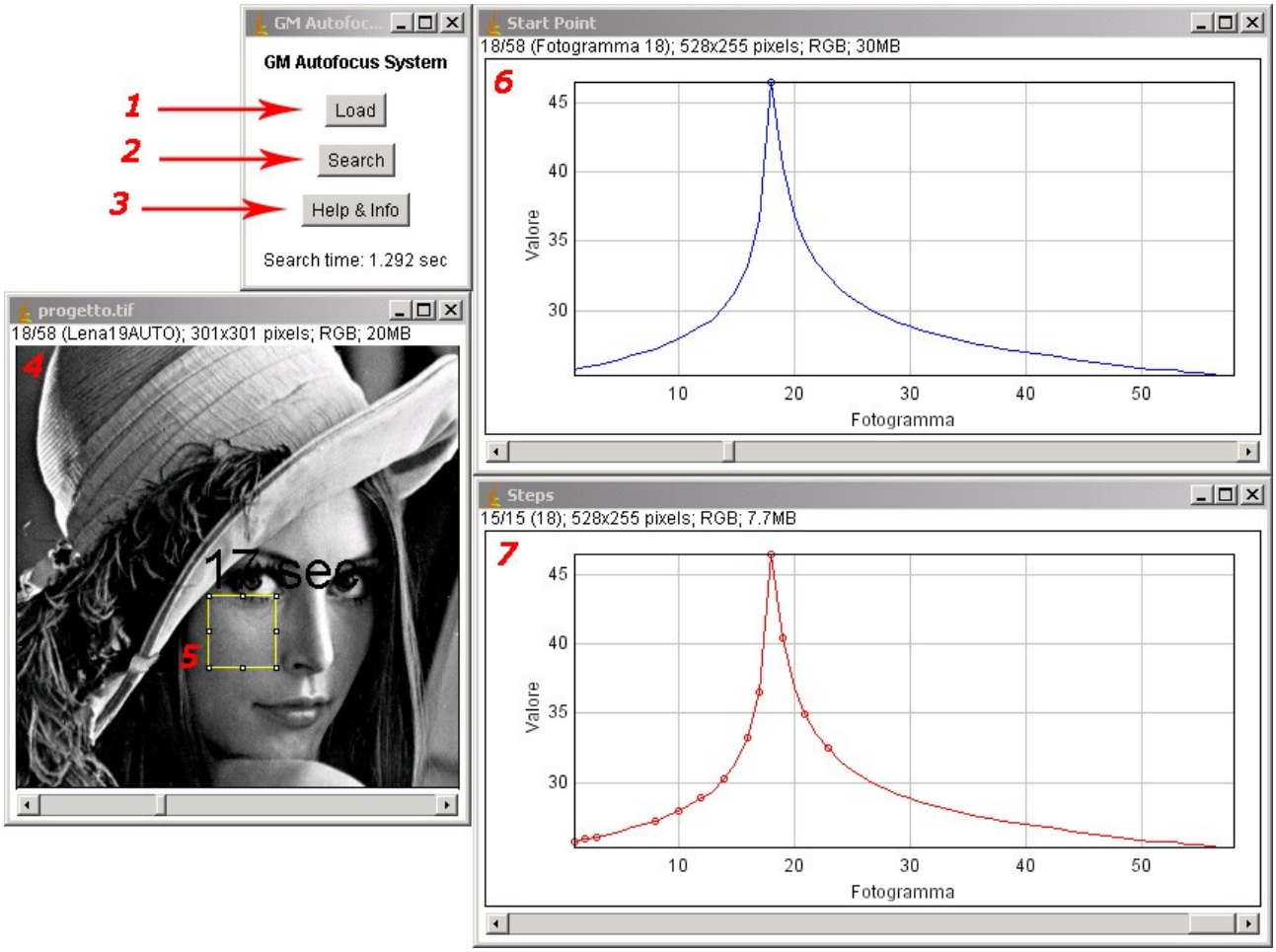

Figura 1

1. **Pulsante "Load"**

   Permette calcolare la funzione di valutazione per tutta la sequenza delle immagini nella sottoimmagine individuata dalla selezione rettangolare 5 presente nella finestra 4. Alla fine del caricamento verrà visualizzata la finestra "Start Point" che permetterà di selezionare il punto da cui far partire la ricerca.

2. **Pulsante "Search"**

   Permette di avviare la ricerca a partire dallo start point selezionato. Alla fine della ricerca verrà visualizzata la finestra "Steps" che permette di visualizzare il fotogramma analizzato ad ogni singolo step e quindi di rendersi conto delle operazioni effettivamente svolte dall'algoritmo di ricerca.

3. **Pulsante “Help&info”**
   Permette di visualizzare una finestra di testo con una breve descrizione del plugin.

4. **Finestra della sequenza di immagini in input**
   Tale finestra contiene lo stack con la sequenza delle immagini su cui effettuare la ricerca. Si può ottenere uno stack del genere per esempio utilizzando il comando File>>Import>>Image Sequence.

5. **Finestra “Start Point”**
   Per tutte le immagini della finestra 4 permette di visualizzare la funzione di valutazione applicata alla sottoimmagine individuata dalla selezione rettangolare 5. Permetterà inoltre di selezionare il punto da cui far partire la ricerca.

6. **Finestra “Steps”**
   Permette di visualizzare la sequenza di fotogrammi analizzati ad ogni singolo step e quindi di rendersi conto delle operazioni effettivamente svolte dall'algoritmo di ricerca. Muovendo il cursore in basso viene aggiornata anche l'immagine visibile nello stack in input.

Una sequenza tipica di operazioni è la seguente:

1. Si apre uno stack o si carica una sequenza di immagini
2. Si definisce una selezione rettangolare
3. Si clicca sul pulsante “Load” che farà visualizzare la finestra “Start Point”
4. Si seleziona il punto da cui far partire la ricerca facendo scorrere lo slider in fondo alla finestra
5. Si clicca sul pulsante “Search” per avviare la ricerca

Vi è un vincolo importane il corretto funzionamento del plugin: ogni volta che si modifica la selezione rettangolare è necessario effettuare un nuovo Load prima di avviare la ricerca.