Biondi Elisa

Zaccaria Damiano

Progetto di Computer Forensics

Plugin per ImageJ: Fourier Elaboration

Corso di Laurea in Informatica.

A.A. 2010-2011

Introduzione

Il progetto realizzato consiste nell'implementazione di un algoritmo di Fourier Elaboration per rimuovere il rumore periodico e le interferenze lavorando sul dominio delle frequenze.

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

Fourier Elaboration

L'elaborazione nel dominio di Fourier permette di selezionare le aree riguardanti le frequenze che si vogliono annullare in modo da ridurre il rumore che queste frequenze causano.

L'implementazione di Fourier Elaboration lavora su qualsiasi immagine in quanto la prima operazione che svolge è quella di applicare la FFT per traslare l'immagine nel dominio delle frequenze. A questo punto l'utente deve selezionare le aree che vuole eliminare cliccando nel punto di inizio della selezione e trascinando il mouse. Al rilascio del mouse, le frequenze selezionate vengono cancellate. Nel caso in cui la selezione non riguardi il centro delle frequenze, saranno annullate anche le frequenze nell'area speculare a quella selezionata. L'utente può effettuare quante modifiche desidera. Nel momento in cui desidera vedere il risultato dell'elaborazione basterà cliccare sul bottone "Apply Inverse FFT" presente nel Fourier Elaboration Panel. L'immagine risultato sarà presentata come una nuova immagine elaborata applicando l'Antitrasformata di Fourier, lasciando quindi a disposizione lo spettro delle frequenze. Nel caso in cui si volessero apportare ulteriori modifiche, basterà quindi tornare sull'immagine dello spettro ed effettuare le ulteriori selezioni desiderate.

Utilizzo del plugin

Per utilizzare il plugin realizzato è necessario installarlo all'interno di ImageJ. Per farlo è necessario inserire nella cartella \ImageJ\plugin\Process i file .class generati dalla compilazione di Fourier_Elaboration_.java. La cartella Process deve essere creata.

Dopo l'avvio di ImageJ, è necessario aprire un'immagine utilizzando la voce 'Apri" dal menù "File".

Nel sottomenu "Process" del menù "Plugins" si troverà la voce "Fourier Elaboration".

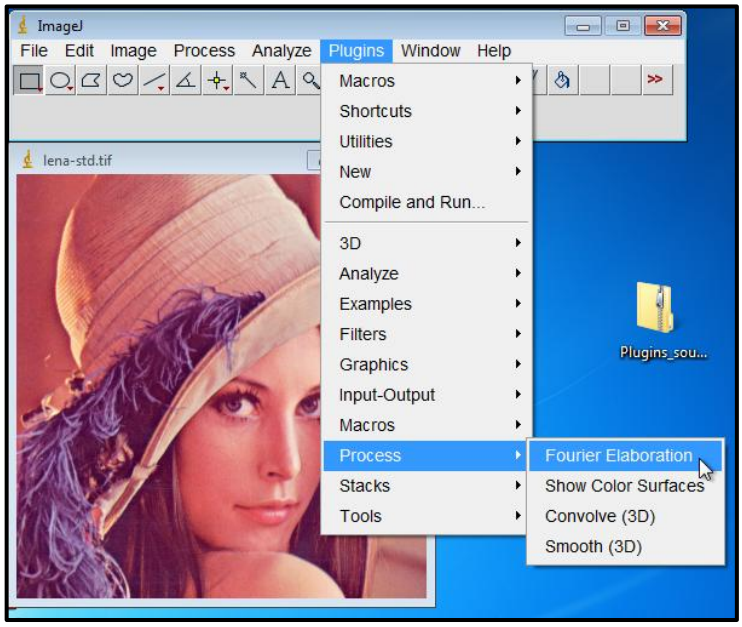

Immagine 1: Posizione del Plugin nel menù

Selezionando tale voce, il plugin verrà avviato sull'immagine aperta. Una volta avviato sarà mostrato all'utente lo spettro delle frequenze dell'immagine su cui dovrà selezionare le aree che riguardano le frequenze da cancellare.

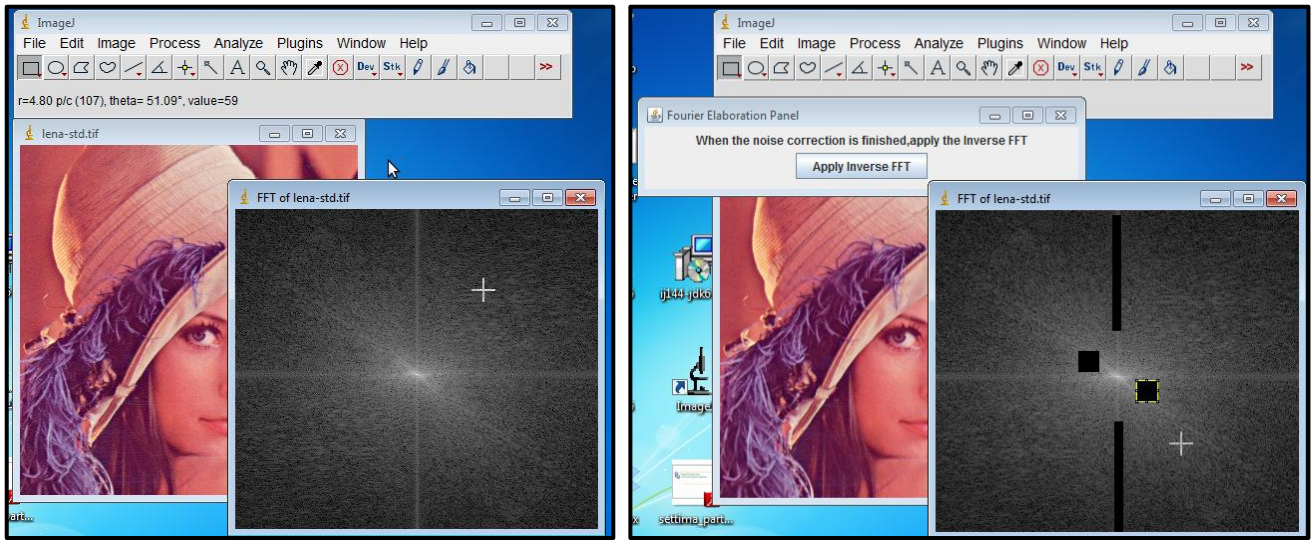

Immagine 2: Spettro delle frequenze e selezioni

Cliccando su “Apply Inverse FFT” nel Fourier Elaboration Panel gli sarà mostrata una nuova immagine contente il risultato dell’elaborazione.

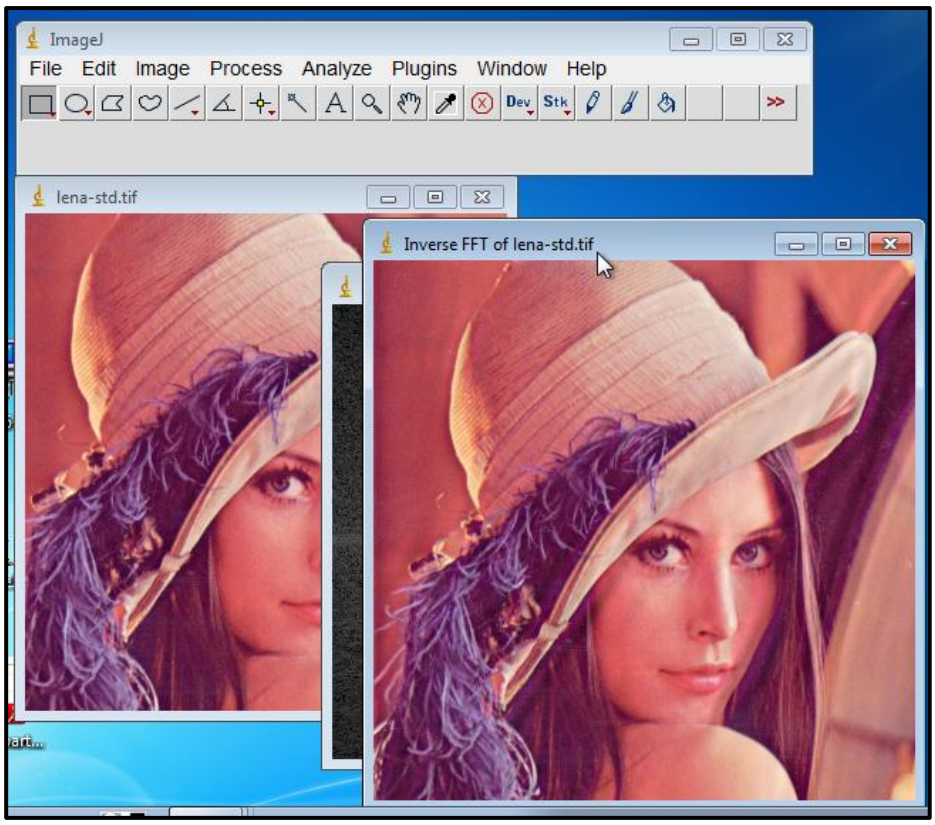

Immagine 3: Risultato dell’elaborazione

Nel caso volesse apportare ulteriori modifiche basterà tornare sull’immagine rappresentante lo spettro delle frequenze e selezionare altre aree da cancellare. Una volta effettuata la prima selezione, ricomparirà il Fourier Elaboration Panel per permettere di richiedere la nuova immagine risultato.

Progettazione

Sono state realizzate due classi.

La classe fftFrame è un’estensione di JFrame che implementa ActionListener. Tale classe è utilizzata per il Fourier Elaboration Panel su cui si trova il JButton per passare dallo spettro delle frequenze all’immagine risultato applicando l’Antitraformata di Fourier.

La classe Fourier_Elaboration_ implementa l'interfaccia PlugInFilter e, dovendo effettuare operazioni basate sulle azioni tramite mouse, l'interfaccia MouseListener.

Per l'interfaccia PlugInFilter, implementa i metodi setup, run e showAbout. Il metodo run applica la Trasformata di Fourier all'immagine attiva (`IJ.run(imp, "FFT", "");`), ne memorizza le dimensioni, la mostra all'utente ed aggiunge come MouseListener per le azioni sulla finestra che lo mostra lo stesso oggetto (`cnvs.addMouseListener(this);`).

Per l'interfaccia MouseListener, implementa vuoti i metodi mouseClicked, mousePressed, mouseEntered e mouseExited in quanto non sono previste azioni relative a questi eventi. Il metodo mouseReleased, eseguito quando viene rilasciato il mouse e quindi al termine di una selezione, crea un oggetto Rectangle per avere a disposizione l'area di interesse selezionata nello spettro delle frequenze e lo elabora richiamando il metodo noise_correction. A questo punto, se il Fourier Elaboration Panel non è già visibile, lo mostra in modo che l'utente possa scegliere di vedere l'immagine risultato. Infine mostra l'immagine dello spettro delle frequenze aggiornato.

```
public void mouseReleased(MouseEvent evt) {
      ImageProcessor spec_ip=spectrum.getProcessor();
      Rectangle roi=spec_ip.getRoi();

      noise_correction((byte[])(spec_ip.getPixels()), roi);

      if(fft.pnl_visibility==false) {
            fft.pnl_visibility=true;
            fft.setVisible(true);
      }

      spectrum.show();
      spectrum.updateAndDraw();
}
```

Il metodo noise_correction ha come parametri di input l'array dei pixel dell'immagine raffigurante lo spettro delle frequenze (`pixs`) e l'area selezionata (`roi`). Il metodo provvede ad azzerare le frequenze selezionate ponendo il valore del pixel nella posizione corrispondente a 0.

```
for(int i=roi.y; i<roi.y+roi.height; i++) {
      int offset=i*width;
      for(int j=roi.x; j<roi.x+roi.width; j++) {
            int pos=offset+j;
            pixs[pos]=0;
      }
}
```

Nel caso in cui la selezione non comprenda il centro, l'azzeramento è applicato anche sui pixel speculari a quelli selezionati (`pixs[pixs.length-pos]=0;`), in quanto rappresentanti le stesse frequenze.

Per capire se il centro appartenga all'area selezionata o meno si utilizza il metodo center_roi. Questo è un metodo booleano che ha come parametro di input l'area selezionata (`roi`). Attraverso una serie di controlli sul pixel di inizio della selezione e sulle dimensioni della stessa, riesce a capire se il centro è stato selezionato o meno e restituisce true o false a seconda del risultato.

```
protected boolean center_roi(Rectangle roi) {
 if((roi.y>height/2) || (roi.x>width/2)) {
      return false;
 }
 elseif((roi.y+roi.height>height/2)&&(roi.x+roi.width>width/2)){
      return true;
 }
 else {
      return false;
 }
}
```