F. CHINARELLI – M.NANNI

SKYRADXY
Un programma per l'identificazione di oggetti sulla DSS

IRA 261/98

INDICE

versione modificata del programma *getimage* fornito dallo *Space Telescope Science Institute*. Il programma originale è stato modificato all'Istituto di Radioastronomia del C.N.R. di Bologna per essere utilizzato via Internet all'interno di *Skyeye* (http://www.ira.bo.cnr.it/skyeye).

La conversione da coordinate equatoriali a pixel eseguita da *Skyradxy* dipende dalla particolare immagine estratta dalla DSS e quindi dai parametri contenuti nell'header fits dell'immagine in esame. *Skyradxy* richede quindi in input il nome del file fits dell'immagine oltre alle coordinate su cui operare le conversione..

L'algoritmo di trasformazione si basa su un sistema di due equazioni in due incognite entrambe elevate fino alla quinta potenza. Poichè tale sistema non è risolvibile analiticamente, per la sua risoluzione si sono dovute utilizzare delle routines matematiche, già disponibili, che si basano su metodi di calcolo numerici. Queste routines e i problemi matematici affrontati con esse verranno descritti nell'ultimo capitolo.

In input il programma legge le coordinate da processare più un'eventuale stringa di commento, mentre in output scrive, per ogni coppia di coordinate lette, la corrispondente posizione (x,y) del pixel sull'immagine, le coordinate stesse e la stringa di commento in caso essa sia stata inserita. Il formato di lettura è libero per rendere più semplice il suo utilizzo come filtro insieme ad altri programmi.

Tutti i problemi ed errori incontrati dal programma in fase di esecuzione vengono registrati nel file *Skyradxy.log*. In ogni caso, a meno che il problema riscontrato non lo renda impossibile, se il programma incontra un errore su una coordinata letta non si arresta: registra il problema in *Skyradxy.log* e chiede le coordinate successive. Le istruzioni per un corretto utilizzo del programma sono sempre scritte in fondo al file *Skyradxy.log*.

Skyradxy è stato scritto in linguaggio C su sisteme operativo Unix, è composto da sette moduli, piu` due header files che vengono inclusi dal preprocessore, ed è suddiviso in diverse funzioni.

Nonostante il programma sia stato realizzato pensando ad un suo impiego all'interno di *Skyeye*, esso può essere utilizzato anche direttamente, nel caso si voglia operare una semplice trasformazione di coordinate. Nel complesso vi sono quattro differenti possibilita' di impiego del programma:

1. lettura da tastiera e scrittura su video
2. lettura da tastiera e scrittura su file
3. lettura da file e scrittura su video
4. lettura da file e scrittura su file

Il primo è il modo più semplice: l'utente inserisce una coppia di coordinate ed il programma restituisce immediatamente su video il risultato. Nel secondo caso invece

declinazione è positiva il segno può essere omesso. Le parentesi quadre indicano che la *stringa* può essere inserita o meno a scelta dell'utente; essa può essere un commento composto da un massimo di 1024 caratteri. Al termine dell'intera riga (caratteri + stringa) va premuto il tasto [*enter*]. Il formato secondo cui immettere le coordinate è libero; vi sono solo alcune regole che l'utente deve rispettare:

- la cifra indicante le ore non può essere preceduta da più di nove caselle di spaziatura;
- i valori numerici devono essere separati tra loro da almeno uno spazio;
- non si possono inserire lettere o caratteri speciali tra le coordinate, sono ammessi solo i segni + o - davanti ai gradi e il punto tra la parte intera e la parte decimale dei secondi d'arco;
- tra le coordinate e la stringa di caratteri vi deve essere almeno uno spazio;

I valori *hh, mm* e *dd* sono dei numeri interi (l'inserimento di un valore reale darebbe un errore) mentre *ss.ss* è reale; essi devono essere compresi tra i valori indicati di seguito:

$$0 \le hh < 24 \quad ; \quad 0 \le mm < 60 \quad ; \quad -90 \le dd \le 90 \quad ; \quad 0.0 \le ss.ss < 60.0 \quad .$$

Se le coordinate non vengono scritte correttamente il programma non si arresta ma, anzichè effettuare la conversione, registra in *Skyradxy.log* l'errore e attende le nuove coordinate.

Al termine della trasformazione il programma scrive sullo schermo rispettivamente la posizione *x,y* del pixel sull'immagine, le coordinate inserite dall'utente e l'eventuale stringa di caratteri, dopodichè attende l'inserimento di nuove coordinate. Se a questo punto si decide di non continuare è sufficiente battere il tasto *enter* per uscire dal programma.

Lettura da tastiera e scrittura su file.

Per lavorare in questo modo si devono inserire nella linea di comando il nome del programma, il nome del file fits, l'eventuale opzione, il carattere " > " e il nome che si vuol dare al file in cui verranno scritti i risultati:

skyradxy nome_file_fits [-p/-s] > nome_file_output

Il carattere " > " si chiama operatore di indirizzamento e serve, in UNIX, a scrivere l'output del programma in un file anzichè sul monitor. Di conseguenza non apparirà nulla sullo schermo e per accedere ai risultati si dovrà aprire il file nome_file_output. L'operatore di indirizzamento non influenza le registrazioni degli errori che appariranno sempre nel file *Skyradxy.log*.

Per quanto riguarda la fase di input resta tutto come descritto nel paragrafo precedente.

e il tasto *enter*. A questo punto vedremo il cursore fermo all'inizio della riga successiva ad indicare che il programma è in attesa di ricevere informazioni. Inseriamo allora le coordinate e il commento:

23 59 20.51 - 28 4 14.30 coordinata di prova

facendo attenzione a battere la prima cifra entro le prime nove colonne della riga e poi terminando col tasto *enter*. A questo punto ci apparirà la linea

659.71 130.96 23 59 20.51 -028 4 14.30 coordinata di prova

dove *659.71* e *130.96* sono le coordinate in pixel sulla sottoimmagine. Il cursore sarà a questo punto fermo all'inizio della riga successiva in attesa di altre coordinate. Se vogliamo continuare inseriamo le nuove coordinate come appena fatto altrimenti battiamo *enter* e usciamo dal programma.

Esempio 2

Immaginiamo ora di avere un file di coordinate che vogliamo trasformare, *coordin*, e di volere scrivere l'output del programma in un file chiamato *coordout*. Supponiamo inoltre di volere utilizzare l'opzione *–s* per evitare il controllo degli errori e che sia sempre *imm1* il nome del file fits contenente l'immagine, mentre il contenuto di *coordin* è:

23 59 20.51 - 28 4 14.30 coordinata di prova

23 59 30.41 - 27 57 59.30

0 0 2.18 -28 1 20.80 ultima coordinata

Inseriamo la riga:

skyradxy imm1 -s < coordin > coordout

il contenuto del file *coordout* sara` il seguente:

659.71 130.96 23 59 20.51 - 28 4 14.30 coordinata di prova

584.00 352.14 23 59 30.41 - 27 57 59.30

335.70 235.01 0 0 2.18 -28 1 20.80 ultima coordinata

Una volta processato il file *coordin,* il programma si arresta.

N.B. – In entrambi gli esempi i risultati ottenuti dipendono anche dalle coordinate equatoriali del centro della sottoimmagine e dalle sue dimensioni. Pertanto le medesime coordinate, in un'altra sottoimmagine, darebbero luogo a risultati differenti.

Descrizione dell'algoritmo

Un immagine digitalizzata è formata da una griglia di pixels ciascuno dei quali è individuato da una coppia di valori *(x,y)*. Tali valori indicano rispetivamente la colonna e la riga occupate dal pixel a partire dall'angolo in basso a sinistra dell'immagine da noi preso

nelle quali α e δ rappresentano rispettivamente ascensione retta e declinazione in radianti mentre h, m, s e g sono nell'ordine ore, minuti, secondi e gradi. In queste espressioni i minuti e i secondi vengono trasformati in frazioni di ore e gradi e sono poi moltiplicati per il fattore $(\pi / 180)$ ottenuto dalla proporzione

$$gradi : 180 = rad : \pi$$

Nell'espressione per il calcolo dell'ascensione retta figura il fattore 15 perchè a π radianti corrispondono 12 ore.

A questo punto si calcolano le coordinate standard χ e ξ in radianti con le formule:

$$\chi = \tan(\alpha - \alpha_c) * (1 - \xi * \tan(\delta_c)) * \cos(\delta_c) , \qquad \xi = \frac{\tan(\delta) - \tan(\delta_c) * \cos(\alpha - \alpha_c)}{\tan(\delta) * \tan(\delta_c) + \cos(\alpha - \alpha_c)} ,$$

dove (α , δ) e (α_c , δ_c) rappresentano le coordinate equatoriali in radianti dell'oggetto e del centro della lastra rispettivamente, e si trasformano in secondi d'arco con le seguenti:

$$\chi = \chi * 206264.806247096 , \qquad \xi = \xi * 206264.806247096 .$$

206264.806247096 sono i secondi d'arco contenuti in un radiante.

Fatto cio` si ricavano le proiezioni, in millimetri, sugli assi x e y della lastra, della distanza tra l'oggetto e il centro di quest'ultima in base alle espressioni:

$$\chi = a_1 x_m + a_2 y_m + a_3 + a_4 x_m^2 + a_5 x_m y_m + a_6 y_m^2 + a_7 \left(x_m^2 + y_m^2\right) + a_8 x_m^3 + a_9 x_m^2 y_m + \\ + a_{10} x_m y_m^2 + a_{11} y_m^3 + a_{12} \left(x_m^3 + x_m y_m^2\right) + a_{13} \left(x_m^5 + 2 x_m^3 y_m^2 + x_m y_m^4\right)$$

$$\xi = b_1 y_m + b_2 x_m + b_3 + b_4 y_m^2 + b_5 x_m y_m + b_6 x_m^2 + b_7 \left(x_m^2 + y_m^2\right) + b_8 y_m^3 + b_9 x_m y_m^2 + \\ + b_{10} x_m^2 y_m + b_{11} x_m^3 + b_{12} \left(y_m^3 + x_m^2 y_m\right) + b_{13} \left(y_m^5 + 2 x_m^2 y_m^3 + x_m^4 y_m\right)$$

nelle quali x_m e y_m rappresentano le proiezioni e a_1, ... , a_{13},, b_1, ... , b_{13} , sono dei coefficienti che servono per la trasformazione e che vengono letti dal file fits. Mettendo assieme queste due espressioni si ottiene un sistema di due equazioni in due incognite di quinto grado. Tale sistema non è risolvibile analiticamente e pertanto viene risolto numericamente sfruttando delle routines di cui si parlerà brevemente nel prossimo capitolo.

principio matematico su cui si basa l'algoritmo per la risoluzione del sistema è un implementazione del metodo di Newton-Raphson per i sistemi di equazioni non lineari. Molto in breve esso si può riassumere nel modo seguente: partendo da un punto prefissato $\boldsymbol{x}_{old} = (x_1, \ldots, x_n)$, si incrementa $\boldsymbol{x}$ di una quantità $\delta\boldsymbol{x}$, ottenendo $\boldsymbol{x}_{new}$ uguale a $\boldsymbol{x}_{old} + \delta\boldsymbol{x}$. Ora si impone la funzione

$$\boldsymbol{F}(\boldsymbol{x}_{new}) = 0 ,$$

dove $\boldsymbol{F} = (F_1, \ldots, F_n)$, e si ottiene così

$$\delta\boldsymbol{x} = -\boldsymbol{J}^{-1} \cdot \boldsymbol{F} ,$$

avendo indicato con $\boldsymbol{J}$ la matrice jacobiana di $\boldsymbol{F}$. Per valutare se accettare o meno l'intero incremento $\delta\boldsymbol{x}$, chiamato step di Newton, si pone

$$f = \frac{1}{2} |\boldsymbol{F}|^2$$

e si verifica che valga la disegualianza

$$f(\boldsymbol{x}_{new}) \leq f(\boldsymbol{x}_{old}) + \alpha \nabla f \cdot (\boldsymbol{x}_{new} - \boldsymbol{x}_{old})$$

in cui $0 < \alpha < 1$. Si noti che, come si può facilmente verificare per sostituzione, il secondo termine del secondo membro della disegualianza è sempre negativo ($\nabla f = \boldsymbol{F} \cdot \boldsymbol{J}$). Il fattore α serve a garantire che f non decresca troppo lentamente in rapporto allo step di Newton, rischiando così di compromettere la convergenza. Esso è stato posto uguale a 10^{-4} come consigliato dagli autori del libro. Se l'ultima disegualianza non è verificata si prende uno step più piccolo moltiplicando $\delta\boldsymbol{x}$ per una quantità λ compresa tra 0 e 1, in cui 1 rappresenta l'intero step di Newton. A questo punto si itera il procedimento finchè i valori della funzione $\boldsymbol{F}$ o di $\delta\boldsymbol{x}$ non soddisfano i criteri di convergenza. Come valori di convergenza sono stati presi quelli indicati dagli autori del libro: 10^{-4} e 10^{-7} rispettivamente per il massimo valore di F_i e di $\delta\boldsymbol{x}$.

Dal punto di vista pratico questo significa che bisogna fornire alla routine *newt.c* i valori iniziali delle incognite, le equazioni che compongono il sistema e la matrice jacobiana ottenuta da queste. Per quanto riguarda λ, essa viene calcolata all'interno di *lnsrch.c* in modo da minimizzare f, ed in ogni caso deve essere compresa tra 0.1 e 0.5.

Anche se in teoria tale procedimento può convergere verso un minimo secondario, costringendo a partire da un altro punto, in pratica questo è molto raro e, nel nostro caso specifico non si è mai verificato anche provando a partire da punti molto esterni al dominio delle due funzioni facenti parte del sistema.

Come valori iniziali delle incognite sono stati presi $x = 0$ e $y = 0$ che corrispondono ad un oggetto che si trovi al centro della lastra. Di conseguenza essi sono mediamente i valori più vicini alla soluzione, compresa, per entrambe le variabili, tra +/- 175 (si ricordi che x e y rappresentano la distanza in millimetri dal centro della lastra e che una lastra misura 35x35 centimetri). Ciò riduce il numero di iterazioni necessarie per convergere al risultato ed il pericolo che i valori delle incognite possano convergere verso un minimo secondario.

noi stimiamo come errore questi valori, in termini angolari esso resta sempre contenuto entro un decimo di secondo d'arco fino a meno di un grado dai poli (probabilmente anche a 20-30 minuti) dove, a causa degli angoli particolari critici i risultati potrebbero essere imprevedibili. Un errore di questa entità rende il programma utilizzabile non solo in *Skyeye* ma anche in buona parte dei possibili problemi astronomici.

Note Finali

Per una descrizione accurata delle routines accennate nel precedente capitolo e dei metodi matematici da esse applicate, si raccomanda la lettura del testo citato. Bisogna precisare infine che per la realizzazione di *skyradxy* si e` partiti da un altro programma, chiamato *xy2rd*, che, come già detto nel precedente paragrafo, svolge l'operazione inversa a *skyradxy*. Esso ha quindi qualche elemento in comune con quest'ultimo soprattutto nella parte riguardante la lettura del file fits, oltre ad utilizzare l'algoritmo che è stato invertito in *skyradxy*.

Bibliografia:

- Flannery et all, "**Numerical Recipes in C**" Cambrige University Press