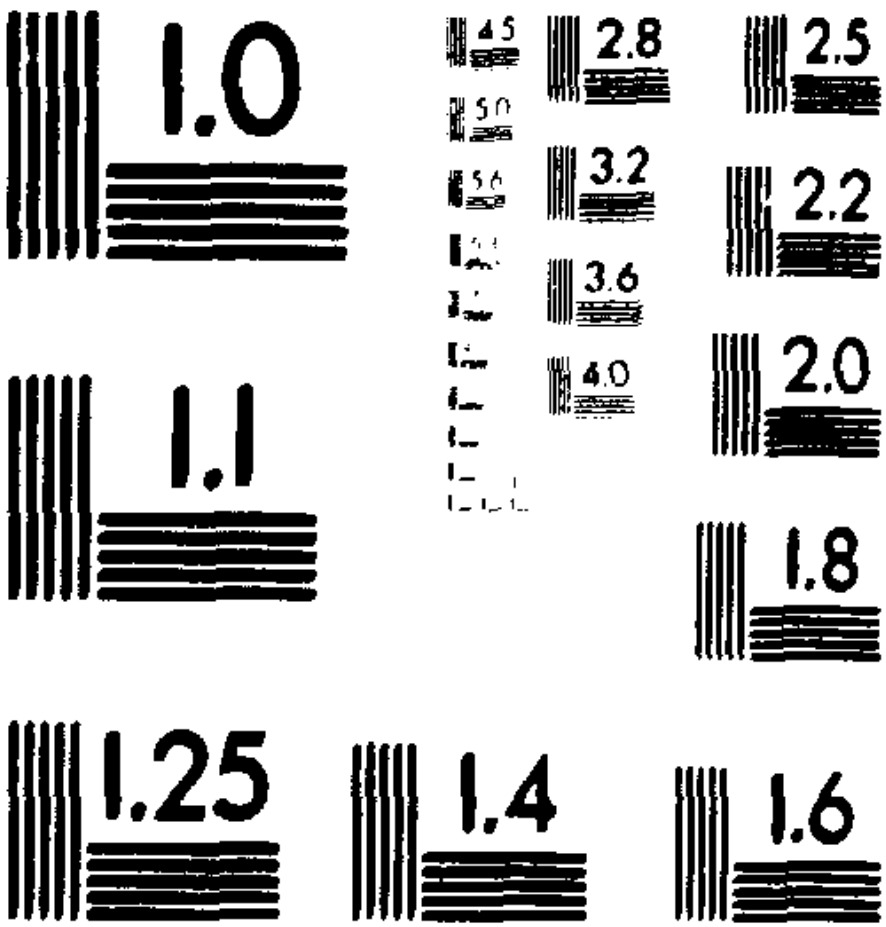
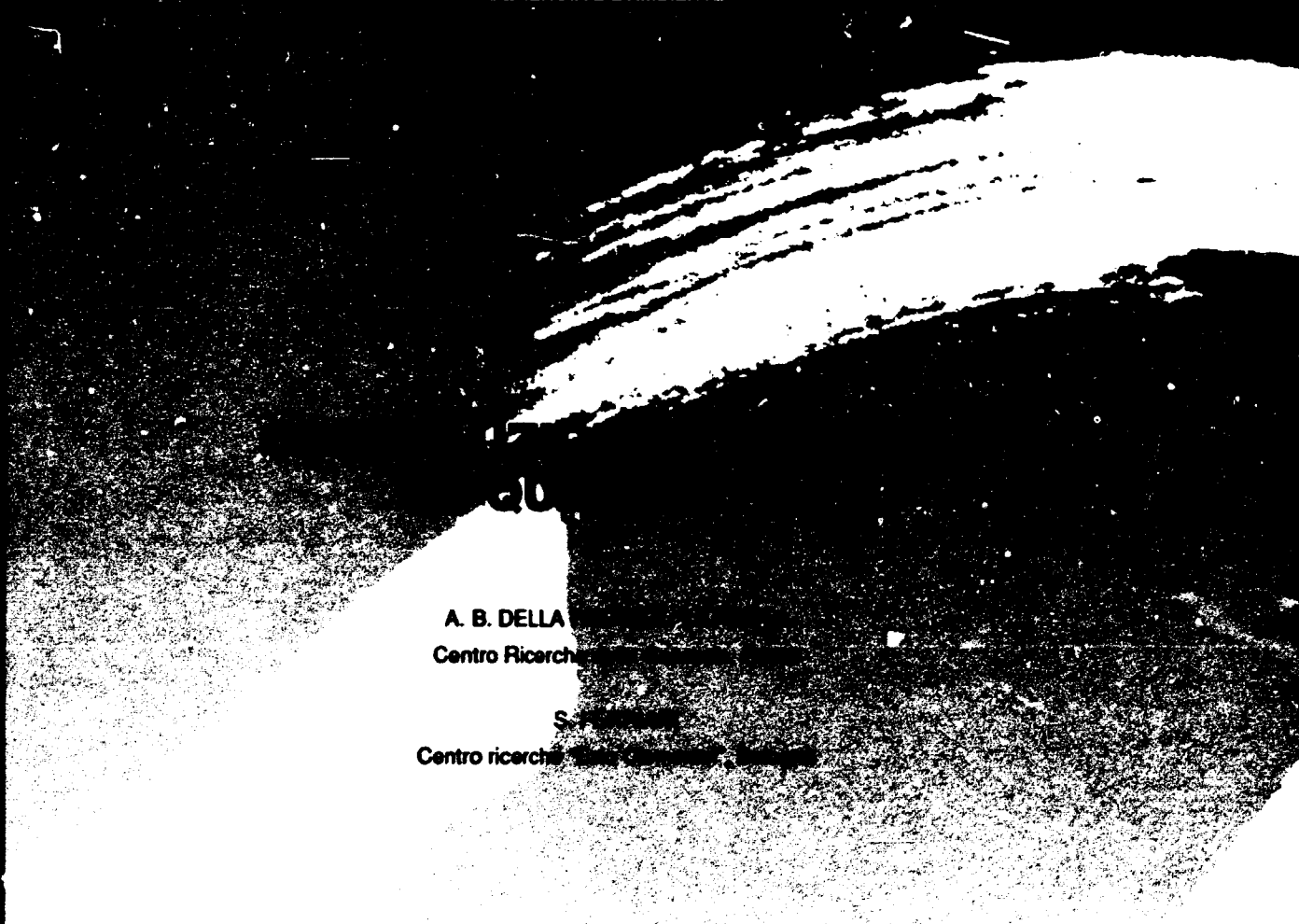


ENEA
ENTE PER LE NUOVE TECNOLOGIE, L'ENERGIA E L'AMBIENTE
ISSN / 1120 - 5571



# [illegible]

A. B. DELLA
Centro Ricerch[illegible]
S.[illegible]
Centro ricerch[illegible]

RT/INN/95/05

ENTE PER LE NUOVE TECNOLOGIE,
L'ENERGIA E L'AMBIENTE

Dipartimento Innovazione

# CONVOLUZIONI DI IMMAGINI SU QUADRICS Q1

A. B. DELLA ROCCA, L. LA PORTA
Centro Ricerche della Casaccia, Roma

S. FERRIANI
Centro ricerche "Ezio Clementel" Bologna

RT/INN/95/05

Testo pervenuto nel maggio 1995

## RIASSUNTO

In questo rapporto viene descritto lo sviluppo e l'implementazione su Quadrics Q1 di un algoritmo per la convoluzione di un'immagine digitale con un kernel, al fine di valutare le prestazioni di tale calcolatore massivamente parallelo nel campo dell'elaborazione delle immagini. Dopo aver richiamato la definizione di convoluzione discreta e le principali caratteristiche h/w e s/w del Q1, viene illustrato l'algoritmo nelle sue varie forme di codifica. Infine, sono riportate le prestazioni ottenute dal Q1 ed il loro confronto con quelle di altre piattaforme hardware. Le considerazioni conclusive mettono in rilievo i risultati e le indicazioni piú importanti.

## *SUMMARY*

*Aimed to evaluate the image processing capabilities of the massively parallel computer Quadrics Q1, a convolution algorithm have been implemented that is described in this report. At first the discrete convolution mathematical definition is recalled together with the main Q1 h/w and s/w features. Then the different codification forms of the algorithm are described and the Q1 performances are compared with those obtained by different computers. Finally, the conclusions report on main results and suggestions.*

# Indice

## 1. Introduzione

Le tecniche di elaborazione delle immagini riconducibili ad algoritmi di convoluzione sono certamente molto numerose e utilizzate per molte applicazioni operative. Basta ricordare, ad esempio, la vasta gamma di filtri numerici che vengono comunemente applicati alle immagini per rimuoverne il contenuto di rumore, o per porne in evidenza particolari aspetti.

Pertanto l'implementazione di un tale algoritmo costituisce un significativo test per valutare le prestazioni nel campo dell'elaborazione delle immagini di una qualunque piattaforma hardware, sia essa ad architettura dedicata o general purpose.

Questa considerazione di carattere generale ha costituito la base delle attivita' descritte nel presente rapporto, il cui obiettivo specifico era di sperimentare le "attitudini", nei confronti dell'Image Processing, del calcolatore massivamente parallelo Quadrics modello Q1.

A tal fine, nel prossimo paragrafo viene brevemente ricordata la definizione di convoluzione discreta tra una immagine ed un kernel in modo da richiamare i presupposti teorici di questo tipo di elaborazione. Fará seguito una descrizione delle caratteristiche essenziali del calcolatore Quadrics Q1, sottolineando quegli aspetti che risultano di particolare interesse. Nei paragrafi 4 e 5 saranno descritti, rispettivamente, l'impostazione generale dell'algoritmo di convoluzione sviluppato e le necessarie operazioni di preprocessing cui va sottoposta l'immagine da elaborare. Il paragrafo 6 é dedicato alla descrizione delle varie strutture di programma con cui é stato implementato l'algoritmo di convoluzione, presentando per ciascuna di esse i risultati ottenuti in termini di tempi di CPU e di riempimento del canale sequenziale di elaborazione (pipe). Infine l'ultimo paragrafo riporta alcune considerazioni conclusive.

## 2. La convoluzione

Le trasformate bidimensionali e le loro proprietà hanno avuto un ruolo importante e continuano ad essere di estremo interesse dal punto di vista delle applicazioni dell'image processing. Esse sono usate in vari settori: per migliorare la qualità delle immagini (enhancement), per ricostruire immagini che siano state degradate da qualche fenomeno fisico (restoration), per comprimere immagini, etc. Inoltre, trasformate quali la convoluzione e la correlazione rivestono particolare importanza perche' sono strettamente legate alla trasformata di Fourier.

Come e' noto [1], la convoluzione di due funzioni f(x,y) e g(x,y), indicata con f(x,y)*g(x,y) é definita dall'integrale:

$$f(x,y)*g(x,y) = \int_{-\infty}^{+\infty}\int f(\alpha,\beta)g(x-\alpha,y-\beta)d\alpha d\beta$$

dove $\alpha$ e $\beta$ sono le variabili di integrazione.

La convoluzione bidimensionale in forma discreta é formulata esprimendo f(x,y) e g(x,y) con due matrici di dimensioni uguali MxN. Pertanto, essa si scrive:

$$f(x,y)*g(x,y)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)g(x-m,y-n)$$

con x=0,1,2....M-1 e y = 0,1,2....,N-1

Invece, la correlazione bidimensionale discreta si scrive:

$$f(x,y)\circ g(x,y)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f^{\oplus}(m,n)g(x+m,y+n)$$

con x=0,1,2....M-1 e y = 0,1,2....,N-1 e $\oplus$ indica il complesso coniugato.

Le relazioni che legano la convoluzione e la correlazione con le trasformate di Fourier di f(x,y) e g(x,y), indicate rispettivamente con F(u,v) e G(u,v), sono le seguenti:

$$\begin{aligned} &f(x,y)*g(x,y) \Leftrightarrow F(u,v)G(u,v) \\ &f(x,y)\,g(x,y) \Leftrightarrow F(u,v)*G(u,v); \end{aligned} \qquad (1)$$

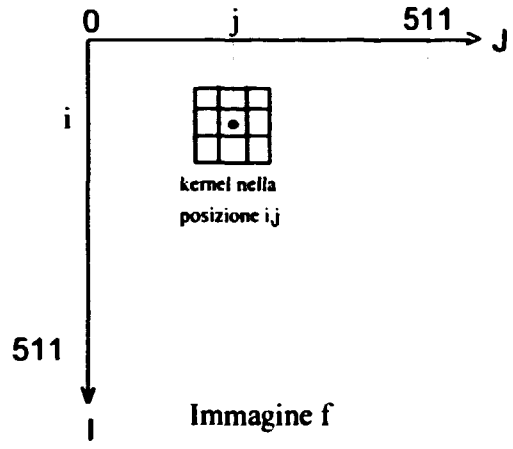
$$\begin{aligned} &f(x,y)^{\circ}\,g(x,y) \Leftrightarrow F^{*}(u,v)G(u,v) \\ &f^{*}(x,y)g(x,y) \Leftrightarrow F(u,v)^{\circ}G(u,v) \end{aligned} \qquad (2)$$

Pertanto, per la (1), effettuare un filtraggio passa basso o passa alto nel dominio delle frequenze é equivalente ad una convoluzione nel dominio dello spazio con una opportuna funzione (o kernel).

Nel seguito del presente rapporto saranno considerate solo immagini costituite da (512x512) pixel e kernel 3x3, poiché costituiscono le dimensioni piú diffuse. In tal caso gli elementi della convoluzione sono schematizzati in Fig.1.

Il valore assunto dalla convoluzione nel generico pixel di coordinate i,j si scrive:

$$\begin{aligned} c(i,j)=\;& w_1\cdot f(i-1,j-1)+w_2\cdot f(i-1,j)+w_3\cdot f(i-1,j+1)+ \\ & w_4\cdot f(i,j-1)\quad +w_5\cdot f(i,j)\quad +w_6\cdot f(i,j+1)\quad + \\ & w_7\cdot f(i+1,j-1)+w_8\cdot f(i+1,j)+w_9\cdot f(i+1,j+1) \end{aligned} \qquad (3)$$




| | | |
|---|---|---|
| $w_1$ (i-1,j-1) | $w_2$ (i-1,j) | $w_3$ (i-1,j+1) |
| $w_4$ (i,j-1) | $w_5$ (i,j) | $w_6$ (i,j+1) |
| $w_7$ (i+1,j-1) | $w_8$ (i+1,j) | $w_9$ (i+1,j+1) |

Kernel centrato nel pixel di coordinate (i,j).
$w_1$.... $w_9$ rappresentano i pesi del kernel, mentre tra parentesi sono riportate le posizioni degli 8 pixel contigui a quello centrale

Fig 1. Schema della convoluzione

L'intera immagine filtrata, a valle del processo di convoluzione, si ottiene facendo scorrere il centro del kernel su tutti i pixel dell'immagine originale e calcolando, per ciascuno di essi, il valore assunto dalla (3). In ciascun pixel, quindi, la convoluzione é frutto della medesima operazione: somma di 9 prodotti; ció che cambia sono solo i fattori dei prodotti. Come meglio si vedrá in seguito, questa situazione é piuttosto favorevole per l'implementazione su computer SIMD (Single Instruction Multiple Data) come il Quadrics e, pertanto, la relazione (3) ha costituito la base fondamentale dell'algoritmo.

## 3. Il Quadrics Q1.

Una descrizione, sia pure sintetica, delle caratteristiche del calcolatore Quadrics ed in particolare del suo modello Q1 é utile per illustrare piú chiaramente le problematiche affrontate nell'implementare l'algoritmo di convoluzione.

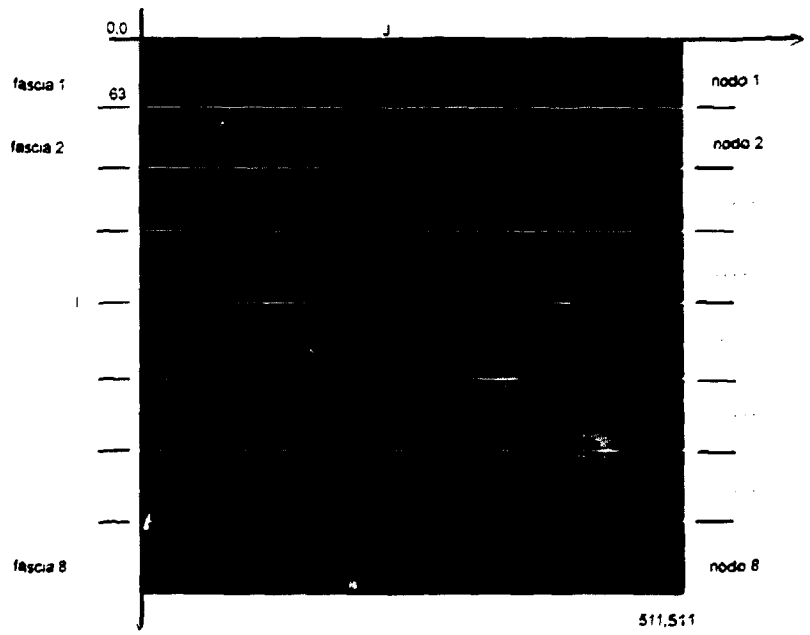
L'architettura hardware di Quadrics [2] é costituita da un insieme di nodi di calcolo interconnessi tra loro a formare una topologia tridimensionale, nel senso che ciascun nodo (PE) puó accedere direttamente solo alle memorie dei suoi immediati vicini posti nelle tre direzioni ortogonali. In particolare, il modello Quadrics Q1 é formato da 8 nodi di calcolo logicamente disposti come i vertici di un cubo ed interconnessi secondo una struttura toroidale. Ciascun nodo é composto da una unita' di calcolo floating-point (MAD), un banco di memoria da 4 Mbyte ed una unitá di Comunicazione e Controllo. A sua volta, ogni MAD contiene 128 registri a 32 bit ed elabora solo dati float in singola precisione con una potenza di picco di 50Mflops grazie anche ad una architettura di elaborazione pipeline. Infine, come tutte le macchine SIMD, i nodi hanno un'unica sincronizzazione che nel Q1 è dettata dalla scheda Z-cpu. L'interazione operativa con Quadrics Q1 avviene tramite un host computer, una workstation SUN, su cui risiede il sistema operativo Quadrics, il compilatore del suo

linguaggio di programmazione TAO ed il caricatore dei programmi.

Il TAO [3] é un linguaggio ad alto livello Fortran-like nel quale sono state introdotte delle capacità strettamente legate alle caratteristiche della macchina Q1. Ad esempio, si ricorda la semplicissima sintassi UP, DOWN, BACK, FRONT, LEFT e RIGHT con cui un nodo puó accedere ai dati posti nella memoria dei nodi vicini. Inoltre TAO contiene comandi di PREPROCESSING, simili a quelli presenti nel linguaggio C, che agiscono durante la sola fase di compilazione. In particolare, risulta molto importante il comando "/fo: k=k1 to k2 { ..statement(k)..}" che ripete, a livello di codice sorgente, l'insieme delle istruzioni ..statement(k).. variando l'indice k dal valore iniziale k1 fino al valore k2. Infine, TAO è dotato dei comandi "extract" e "replace". Il primo consente di prelevare interi blocchi di dati dalla memoria depositandoli nei registri del MAD, viceversa il secondo effettua l'operazione opposta. Il loro impiego congiunto ottimizza le operazioni di calcolo, riducendone significativamente i tempi complessivi.

## 4. Impostazione dell'algoritmo

Concludendo il paragrafo 2 si é constatato che il processo convolutivo comporta l'applicazione della medesima espressione algebrica (relazione 3) su tutti i pixel di una immagine. Pertanto, nella sua generalitá, l'algoritmo da implementare su Q1 é stato impostato sulla ripartizione del calcolo tra gli 8 nodi, affidando a ciascuno di essi una specifica fascia dell'immagine ed avendo cura che ciascuna fascia fosse corredata di una zona di sovrapposizione, pari ad una linea (overlay), con la fascia precedente e la successiva (Fig. 2) [4].




F.g. 2 Frazionamento dell'immagine da elaborare in 8 fasce

Quest'ultimo accorgimento rende pienamente autonomo ogni nodo, eliminando la necessitá di accesso ai pixel posti nelle memorie dei nodi ad esso connessi.

Con tale approccio, il tempo di calcolo per effettuare la convoluzione su tutta l'immagine corrisponde al tempo necessario al singolo MAD per effettuare la convoluzione sulla rispettiva fascia ampia (64 x 512) pixel.

## 5. Operazioni di preprocessing.

Al fine di caricare i dati di ciascuna fascia nella memoria del rispettivo nodo é stato necessario, tuttavia, sottoporre preventivamente l'immagine ad un processo di pre-elaborazione avente i seguenti obiettivi:
- convertire i valori interi dei pixel (1 Byte/pixel), esprimendoli in formato reale in singola precisione (4 Byte/pixel);
- aggiungere ad ogni fascia le rispettive zone di overlay;
- memorizzare tutti i dati per tutti i nodi in un unico file. Ciò é necessario poiché non é possibile scambiare direttamente dati tra la memoria dell'host e la memoria dei nodi.

Per conseguire i predetti obiettivi, é stato implementato su SUN uno specifico modulo software che, fissata la dimensione del kernel (3x3), trasforma il file contenente l'immagine da sottoporre a convoluzione in un nuovo file avente la sequenza ordinata di 8 fasce da 64+2 righe ciascuna.

## 6. Implementazione dell'algoritmo

Alla luce di quanto esposto nei due paragrafi precedenti il programma di convoluzione implementato su Q1 é stato articolato nei seguenti passi principali:
a) lettura sequenziale del file ottenuto a valle della pre-elaborazione e smistamento dei dati immagine nei rispettivi nodi;
b) computo della convoluzione da parte di ogni MAD;
c) scrittura in un unico file, e nella stessa sequenza utilizzata in fase di lettura, dei risultati ottenuti da ogni nodo. Questo file contiene l'immagine filtrata che, a valle di un processo di post-elaborazione, viene visualizzata sul monitor dell'host. Nella fattispecie, la Fig.3 illustra il risultato di un filtraggio passa-alto applicato all'immagine di Fig. 2.

La codifica in linguaggio TAO dei suddetti passi non ha presentato particolari problemi per le azioni incluse in a) e c), in quanto le funzioni di I/O sono gia' immediatamente disponibili nel linguaggio.

Fig 3. Risultato della convoluzione con il kernel 3x3

Una maggiore attenzione ha invece richiesto il passo b). Esso, infatti, è stato codificato in diverse forme, il cui susseguirsi rispondeva all'esigenza di migliorare via via le prestazioni di calcolo.

La forma di codifica più intuitiva ed immediata (rivelatasi in seguito anche la piú primitiva) ha portato alla struttura di programma racchiusa nel seguente schema:

```
DO i = 1 , 64
   DO j = 1 , 512
      DO k = 1 , 3
         DO l = 1 , 3
            c(i,j) = c(i,j) + f(i-k,j-l) * kernel(k,l)
         END DO
      END DO
   END DO
END DO
```

dove c(i,j) é il valore computato della convoluzione, f(i,j) contiene i dati immagine ed infine kernel(k,l) esprime i pesi del filtro. I primi due cicli DO servono per far scorrere il kernel su tutte le righe (64) e su tutte le colonne (512), mentre i cicli piú interni servono a calcolare la relazione (3) ed a memorizzarne il risultato nella matrice c(i,j). Con questa struttura del programma si é ottenuto un tempo di CPU pari a 2640 msec ed un riempimento della pipe dello 0%. É utile sottolineare che, per questa struttura come per le successive, il tempo di CPU si riferisce al solo passo di computazione, esso non include pertanto i tempi di I/O.

Al fine di diminuire il tempo di calcolo, é stata utilizzata l'istruzione "/for " per operare un sviluppo dei cicli (unrolling dei loop). La conseguente struttura del programma è sintetizzata

nel seguente schema:

```
DO i = 1 , 64
   DO j = 1 , 512
      /for  k = 1  to  3 {
         /for  l = 1  to  3 {
            c(i,j) = c(i,j) + f(i-k,j-l)* kernel(k,l)
         }
      }
   END DO
END DO
```

L'inserimento dell'istruzione "/for", ha consentito al compilatore TAO di aumentare il riempimento della pipe. In tal modo, i MAD effettuano un maggior numero di operazioni elementari nello stesso tempo. Questa struttura ha portato il tempo di calcolo a 640 msec, con una riduzione del 76% rispetto al caso precedente, ed un coefficiente di riempimento pipe dell'1%.

La forma di codifica successiva ha visto l'inserimento delle istruzioni TAO matrix, extract e replace, pervenendo alla seguente struttura di programma:

```
DO  i = 1 , 64
   DO  j = 1 , 512, 32
      extract a1 from kernel
      /for n = 1 to 32 {
         /for  k = 1  to  3 {
            extract a2 from immagine
            /for  l = 1  to  3 {
               somma = somma + a1.[k,l]*a2.[l]
            }
         }
      out.[n] = somma
      }
      replace out  into  risultato
   END DO
END DO
```

Il primo "extract" trasferisce i 9 pesi del kernel sui registri del MAD dove vengono indirizzati tramite la variabile a1, mentre il secondo "extract" carica i dati immagine f(i,j), 3 per volta, in altri registri del MAD dove vengono indicati con il nome simbolico a2. I due "/for" più interni computano la convoluzione. La variabile out memorizza temporaneamente, sempre sui registri del MAD, i risultati della convoluzione, che vengono poi trasferiti in blocchi di 32 in memoria

tramite il comando "replace".

Il tempo impiegato dal Q1 é sceso a 197 msec con una riduzione, rispetto al caso precedente, del 36% ed un riempimento della pipe pari al 5% .

Per ottenere ulteriori diminuzioni del tempo di calcolo, anzichè proseguire sulla strada dell'affinamento della struttura del programma, si é agito piuttosto sulla disposizione dei dati immagine nella memoria dei singoli MAD.

In questa ottica, alla fase di preprocessing descritta nel paragrafo 5, è stata aggiunta una nuova azione col seguente obiettivo specifico: comporre, per ogni pixel f(i,j) dell'immagine, un vettore V(k),k=1,9 che raggruppasse il pixel stesso ed i suoi 8 immediati vicini e che fosse ordinato a partire da quello posto in alto a sinistra. Per ogni coppia di coordinate (i,j), gli elementi del vettore V(k) sono, pertanto, i seguenti:

V(1)=f(i-1,j-1); V(2)=f(i-1,j); V(3)=f(i-1,j+1)
V(4)=f(i,j-1); V(5)=f(i,j); V(6)=f(i,j+1)
V(7)=f(i+1,j-1); V(8)=f(i+1,j); V(9)=f(i+1,j+1)

Si definisce in tal modo una sorta di meta-immagine, nel senso che essa rappresenta uno stadio intermedio di trasformazione dell'immagine da elaborare, utile per pervenire allo stato finale di immagine filtrata. Concettualmente la meta immagine è formata da 9 piani ed ha la struttura tridimensionale illustrata in Fig. 4.

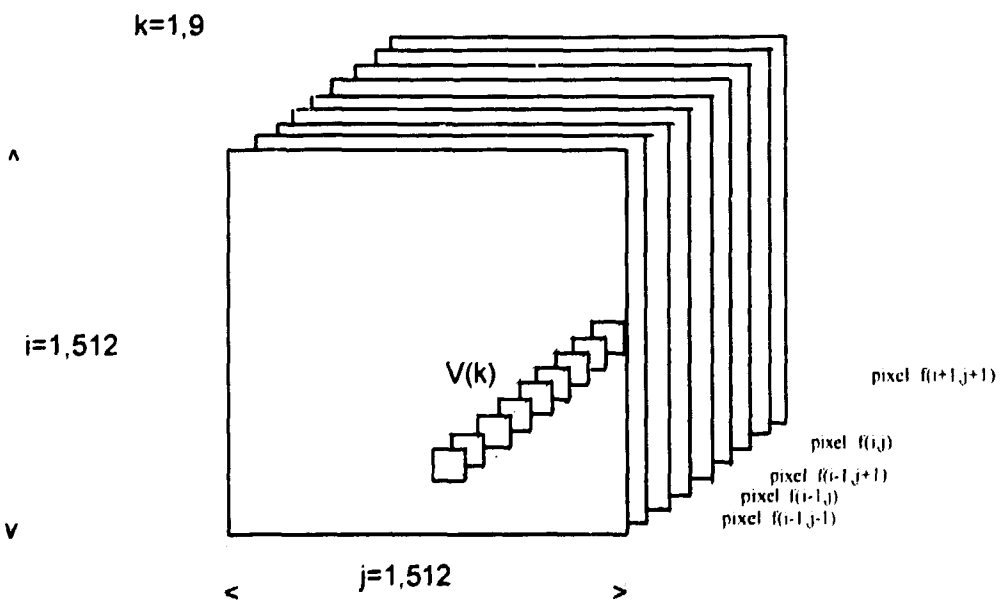



Fig. 4 Struttura 3-D della meta immagine

Con tale disposizione dei dati, il programma TAO ha assunto la seguente struttura:

```
DO i = 1 , 64
   DO j = 1 , 512*9 , 9
        extract a1 from kernel
        extract V from meta-immagine
             /for k = 1 to 9 {
                  somma = somma + a1.[k]*V.[k]
             }
     END DO
  END DO
```

Il primo "extract", similmente al caso precedente, trasferisce i 9 valori del kernel sui registri del MAD dove sono indirizzati tramite la variabile a1. Il secondo "extract" carica, su altri registri del MAD dove vengono indicati con la variabile V, i 9 pixel dell'immagine necessari al computo del valore corrente della convoluzione. Va notata la differenza con la struttura precedente, in cui i pixel erano caricati 3 alla volta. Nella struttura sopra illustrata, non sono stati esplicitati, per maggiore chiarezza, la logica del "replace" e dell'unrolling del loop del tutto simili al caso precedente.
Con tale accorgimento (che comporta chiaramente una maggiore occupazione di memoria) si é ottenuto un riempimento della pipe del MAD pari al 18% e una riduzione del tempo di calcolo di circa il 71% per un ammontare complessivo pari a 58 msec.

A questo punto l'obiettivo principale è diventato quello di individuare forme di codifica che migliorassero ulteriormente l'utilizzo della pipe. Ciò è stato conseguito con la seguente struttura di programma:

```
  DO i = 1 , 64
     DO j = 1 , (512-32)*9, 32*9
          extract a1 from kernel
          /for n = 1 to 32 step 4 {
             extract a2 from immagine
/for l = 0  to  8  { out.[n]   = out.[n]   + a1.[l]   *a2.[l]  }
/for l = 9  to  17 { out.[n+1] = out.[n+1] + a1.[l-9] *a2.[l]  }
/for l = 18 to 26 { out.[n+2] = out.[n+2] + a1.[l-18]*a2.[l]  }
/for l = 27 to 35 { out.[n+3] = out.[n+3] + a1.[l-27]*a2.[l]  }
             }
      replace out into risultato
      END DO
   END DO
```

Il primo "extract" memorizza sui registri i pesi del kernel dove sono indicati con la variabile a1. Il secondo extract trasferisce 9x4=36 dati immagine sui registri indirizzandoli con la variabile a2. Viene quindi calcolato il valore della convoluzione in 4 pixel successivi. E' da notare che, nel caso precedente questo calcolo veniva effettuato un pixel alla volta. Infine, si scaricano in memoria i 32 valori calcolati tramite il comando "replace". Con questa struttura di programma si è ottenuto un tempo di CPU pari a 31 msec. ed un riempimento della pipe del 35%.

Tutti i risultati precedenti sono riassunti nella tabella 1:

| STRUTTURE DI PROGRAMMA | CPU (msec) | DECREM (%) | PIPE (%) |
|---|---|---|---|
| **Programma iniziale** | **2640** | | **0** |
| **Sostituzione cicli "DO" con "/for"** | **640** | **76** | **1** |
| **Inserimento "extract" e "replace"** | **197** | **69** | **5** |
| **Nuova disposizione dei dati nei MAD** | **58** | **71** | **18** |
| **Calcolo con passo 4** | **31** | **46** | **35** |
| **Tempo limite Quadrics Q1** | **11** | | |

Tab. 1 Prestazioni Quadrics Q1

L'ultima riga della Tab. 1 si riferisce al tempo minimo di calcolo necessario al Q1 per effettuare la convoluzione. Infatti, ricordando che la potenza di picco ammonta a 8x50MFlops e che sono necessarie 18 operazioni per calcolare il singolo valore di convoluzione, virtualmente il tempo minimo è dato da:

$$(512*512*18)/400\text{MFlops} = 11 \text{ msec}$$

La Tab.1 rispecchia i progressi fatti nel programmare il Q1 evidenziandoli con il miglioramento graduale delle prestazioni ottenute in termini di tempo di CPU. Tuttavia, tali prestazioni conservano ancora un carattere relativo, poiché manca un termine di paragone che esprima le prestazioni di altre piattaforme.

Pertanto, volendo conferire un significato più ampio a quanto ottenuto dal Q1, é stato implementato il medesimo algoritmo di convoluzione su altri computer disponibili e precisamente: Pentium a 60 MHz, VAX ALPHA ed un sistema ad architettura dedicata all'Image Processing che include una scheda "Real Time Convolver" (RTC). Quest'ultima opera in "real time" nel senso che effettua la convoluzione di una immagine (512x512)pixel x 8bit/pixel con un kernel 3x3 nel tempo di acquisizione di una singola frame da un segnale video standard (1/25 sec pari a 40 msec). Pertanto, tale prestazione puó essere intes. come

l'attuale limite tecnologico, almeno sul piano dei sistemi di calcolo piú ricorrenti nel trattamento immagini.

Le prestazione ottenute sono riassunte nella Tab. 2:

| PIATTAFORME HARDWARE | CPU (msec) |
|---|---|
| PENTIUM 60 MHz | 880 |
| VAX ALPHA | 210 |
| REAL TIME CONVOLVER | 40 |

Tab. 2 Prestazioni di altre piattaforme

Naturalmente il confronto diretto di tale risultati con quelli della Tab. 1 non é omogeneo, anche perché nessuna di queste piattaforme ha una architettura massivamente parallela. Tuttavia, la comparazione risulta molto utile per dare un inquadramento generale delle prestazioni ottenute con il Q1. Numerose considerazioni potrebbero farsi, ma si evidenzia chiaramente che il minor tempo di calcolo, 31 msec, è stato ottenuto con il Quadrics Q1.

Tale valore, confrontato con i 40 msec del RTC acquista un particolare significato se si tiene presente che quest'ultimo opera su grandezze espresse con 8 bit, mentre il Q1 elabora grandezze reali a 32 bit.

## 7. Conclusioni

L'obiettivo principale delle attivitá descritte nei paragrafi precedenti era di sperimentare le attitudini del Quadrics Q1 verso l'Image Processing. I risultati raggiunti ed i confronti fatti in tal senso consentono certamente di giudicarle ottime sul piano delle capacitá computazionali, tenendo anche conto dei seguenti fattori: - Q1 non ha una architettura appositamente progettata per l'Image Processing; - ci sono ancora margini di miglioramento come indica il raffronto in Tab. 1 tra i 31 msec ottenuti ed il tempo minimo virtuale di 11 msec.

Purtroppo, non puó essere espressa la medesima valutazione per quanto concerne le capacitá di I/O che, allo stato attuale, rimangono ben al di sotto delle esigenze applicative. I tempi necessari per caricare o restituire l'immagine digitale all'host risentono notevolmente della bassa velocita' di I/O che, per la versione utilizzata del sistema operativo del Quadrics, e' dell'ordine di 700 kByte/sec.

Oltre a questi elementi di giudizio le attivitá hanno fatto emergere alcune indicazioni che vale la pena di riportare, dal momento che il loro significato supera i confini del contesto specialistico in cui sono state maturate. Tali indicazioni riguardano in particolare:
- l'importanza della struttura del programma in linguaggio TAO. Come si é visto, uno dei fattori decisivi per migliorare le prestazioni del Q1 é stato quello di aumentare il riempimento della pipe sia utilizzando le istruzioni TAO piú adatte ("extract", "replace", etc.), sia

riorganizzando i dati in memoria. La forte incidenza di questo fattore sulle prestazioni risulta evidente scorrendo la colonna delle riduzioni percentuali dei tempi (DECREM) riportata in Tab. 1;

- la grande influenza della disposizione dei dati nella memoria dei MAD. Tale disposizione puó anche non rispettare la configurazione con cui i dati sono disposti nella realtá fisica che essi rappresentano, l'importante é che la loro ristrutturazione sia funzionale agli obiettivi di elaborazione che si vogliono conseguire. Questa indicazione sembra sottolineare, a sua volta, l'utilitá di corredare Quadrics con un sistema di I/O che, al tempo stesso, risulti sufficientemente veloce e sia dotato di capacitá autonome con cui operare sui dati in corso di acquisizione e/o di restituzione.

BIBLIOGRAFIA

[1] Rafael C. Gonzalez , Paul Wintz. Digital Image Processing. Second Edition. Addison-Wesley Publishing Company 1987

[2] Quadrics Operating System. Versione α. Alenia Spazio SpA 1993.

[3] The TAO Language. Versione α. Alenia Spazio SpA 1993.

[4] AA.VV. La chimera d'Arezzo. ENEA 1992

Edito dall'Enea
Direzione Relazioni Esterne
V. le Regina Margherita, 125 - 00198 Roma
Finito di stampare nel mese di maggio 1995
presso il Tecnografico