Programmazione Parallela e Funzionale

Corsi di Laurea in Ingegneria Informatica e Automatica, Ingegneria dei Sistemi Informatici, e Laurea Magistrale in Ingegneria Informatica

Sapienza Università di Roma

Esercizio: raddoppiamento delle dimensioni di un'immagine

Lo scopo dell'esercizio è quella di scrivere un modulo C che, data in input una immagine a toni 256 di grigio di dimensione $w \times h$, crei una nuova immagine allocata dinamicamente ottenuta da quella di input raddoppiandone altezza e larghezza, come nell'esempio sotto.

(a) Immagine originale a 256 toni di grigio di dimensione 500×334

(b) Immagine raddoppiata di dimensione 1000×668

Si vada nella directory di lavoro `resize2x` e si definisca nel file `resize2x.c` la funzione `resize2x` con il seguente prototipo:

```
void resize2x(unsigned char* in, int w, int h,
              unsigned char** out, int* ow, int* oh,
              clut_device* dev, double* td);
```

dove:

- `in`: puntatore a un buffer di dimensione `w*h*sizeof(unsigned char)` byte nella memoria dell'host che contiene l'immagine di input in formato row-major;
- `w`: larghezza di `in` in pixel (numero di colonne della matrice di pixel);
- `h`: altezza di `in` in pixel (numero di righe della matrice di pixel);
- `out`: parametro in cui restituire il puntatore all'immagine di output in formato row-major, **che la funzione deve allocare dinamicamente al suo interno**;
- `ow`: parametro in cui restituire la larghezza dell'immagine di output in pixel;
- `oh`: parametro in cui restituire l'altezza dell'immagine di output in pixel;
- `dev`: ambiente di esecuzione della GPU (si veda `clut.h`);
- `td`: parametro in cui restituire la durata dell'esecuzione del kernel.

Scrivere un opportuno kernel OpenCL nel file `resize2x.cl`.

Suggerimento: eseguire il kernel su un NDRange a due dimensioni dove la dimensione 0 corrisponde all'asse orizzontale (x) e la dimensione 1 corrisponde all'asse verticale (y).

L'immagine deve essere riscalata mediante interpolazione lineare facendo la media dei toni di grigio di pixel adiacenti come nel seguente esempio, dove la matrice di input è 3×3 e quella di output è 6×6 (sui bordi si usino i valori di grigio originali):

a	**b**	**c**
d	**e**	**f**
g	**h**	**i**

Immagine input

a	*(a+b)/2*	**b**	*(b+c)/2*	**c**	*c*
(a+d)/2	*(a+e)/2*	*(b+e)/2*	*(b+f)/2*	*(c+f)/2*	*c*
d	*(d+e)/2*	**e**	*(e+f)/2*	**f**	*f*
(d+g)/2	*(d+h)/2*	*(e+h)/2*	*(e+i)/2*	*(f+i)/2*	*f*
g	*(g+h)/2*	**h**	*(h+i)/2*	**i**	*i*
g	*g*	*h*	*h*	*i*	*i*

Immagine output

Compilazione e test.

Directory di lavoro: `resize2x/`

1. Compilazione programma di test (*una tantum*): dare il comando `make`, che genera il file eseguibile `resize2x`
2. Compilazione ed esecuzione kernel:
   a. `make test1`: test su immagine 500×334 a 256 toni di grigio del Colosseo;
   b. `make test2`: test su una versione 512×512 a 256 toni di grigio del classico bechmark "Lena" usato in computer graphics.

Nella directory `results` verranno generate varie immagini ottenute aggiungendo diverse cornici alle immagini date in input. Le immagini, salvate in formato PGM (Portable Graymap Format), possono essere visualizzate con il programma Gimp.