Il sottocampionamento

Il sottocampionamento, noto anche come subsampling, è un algoritmo che permette di ridurre il numero di campioni di un segnale digitale oppure di un'immagine senza andare incontro, durante l'inevitabile perdita di informazione, al famoso difetto di aliasing. E' importante ridurre il più possibile questo difetto, perchè l'aliasing potrebbe provocare nei suoni un fastidioso brusio oppure nelle immagini una percezione di forme imprecise o addirittura totalmente alterate.

- Il sottocampionamento dei segnali digitali

Per capire meglio di cosa stiamo parlando, riprendiamo velocemente qualche cenno di segnali digitali. Un segnale digitale è una raccolta di n campioni, tipicamente campionati da un segnale analogico in base a un tempo di campionamento (durata temporale del campione), quindi si ha una frequenza di campionamento che corrisponde al numero massimo di campioni riprodotti al secondo (es: 44000). Il teorema del campionamento dice che la massima frequenza di campionamento (e quindi il numero di campioni) è scelta come circa il doppio della frequenza di taglio del segnale analogico da campionare, per esempio un suono con massimo 20000 Hz di banda passante (udibile umano) può essere campionato usando 48000 campioni al secondo per ottenere una qualità perfetta durante la fase di ascolto (ossia durante la conversione digitale-analogico). Tuttavia se riduciamo il numero di campioni da 48000 a 12800 al secondo otteniamo un'irrimediabile perdita di informazione ed un conseguente effetto di aliasing, che all'orecchio ci suonerà come un fastidioso brusio o altri tipi di rumore:

```
prima = 48000; dopo = 12800;
step = prima / dopo; p = 0;

For n = 0  to dopo-1
        segnale_nuovo[n] = segnale_vecchio[int(p)];
        p += step;
Next
```

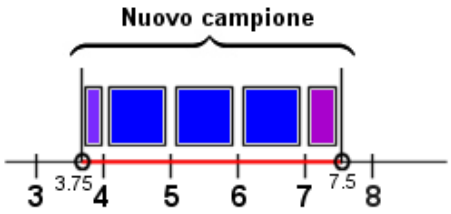
Con questo algoritmo, noto anche come stretching, abbiamo costruito un nuovo segnale digitale sulla base di quello vecchio, che era da ridurre. Il nuovo segnale digitale ha preso un campione da quello vecchio, poi ne ha incrementato la posizione di lettura di 3.75 per poi prenderne un altro, questo significa che almeno 2 o 3 campioni sono stati scartati per ogni ciclo. Vediamo numericamente come si comporta la posizione di lettura p (che è un numero in virgola) e la posizione effettiva del campione da leggere (che è un numero intero).

```
n = 0;   p = 0;       int(p) = 0;
n = 1;   p = 3.75;    int(p) = 3;    'scartati 1, 2
n = 2;   p = 7.5;     int(p) = 7;    'scartati 4, 5, 6
n = 3;   p = 11.25;   int(p) = 11;   'scartati 8, 9, 10
```

n = 4; p = 15; int(p) = 15; 'scartati 12, 13, 14
n = 5; p = 18.75; int(p) = 18; 'scartati 16, 17

Il sottocampionamento di un segnale consiste proprio nell'evitare di scartare questi campioni ed attribuire il loro peso informativo al nuovo campione sotto forma di media pesata per evitare l'effetto di aliasing. Il peso da associare ad ogni campione viene stabilito in base alla sua dimensione (misurata da 0 e 1) e prevedendo quanti campioni verrebbero scartati dopo un incremento della posizione di lettura. Vediamo come dovrebbe essere calcolato il nuovo campione, nel caso di n = 1:

p1 = 3.75; int(p1) = 3; peso1 = 4 – 3.75 = 0.25;
p2 = 4; int(p2) = 4; peso2 = 5 - 4 = 1;
p3 = 5; int(p3) = 5; peso3 = 6 - 5 = 1;
p4 = 6; int(p4) = 6; peso4 = 7 - 6 = 1;
p5 = 7.5; int(p5) = 7; peso5 = 7.5 - 7 = 0.5;

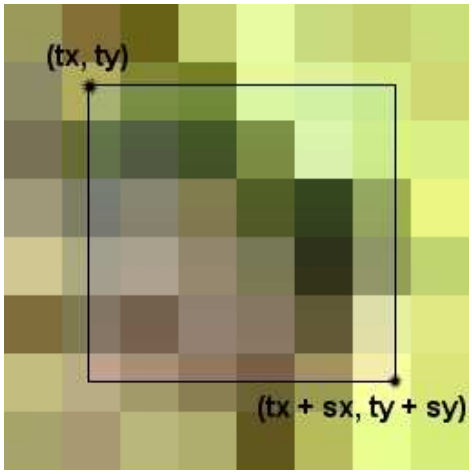


Il campione nella posizione 3 ha un peso informativo di 4-3.75 = 0.25. Poi in ordine viene letto il campione successivo della posizione 4 con un peso di 1, e così via, fino ad arrivare alla posizione 7 con un peso di 0.5. Possiamo dunque renderci conto che il nuovo campione dovrà essere uguale alla media pesata di ben 5 campioni (la somma del campione di partenza, i 3 che sarebbero stati scartati e l'ultimo di arrivo). Indichiamo con s[n] il vecchio segnale e calcoliamo il nuovo campione nel seguente modo: nuovo_campione[1] = (s[3]*0.25 + s[4]*1 + s[5]*1 + s[6]*1 + s[7]*0.5) / 3.75; Osservando questa formula possiamo notare che gli unici campioni da moltiplicare per i pesi sono quello di partenza e quello di arrivo, mentre gli altri andrebbero moltiplicati per 1, cioè andrebbero sommati per come si presentano. Inoltre la somma dei pesi è pari alla lunghezza massima del campione, che in questo caso è pari allo step: possiamo creare dunque un algoritmo ottimizzato, abbastanza leggero dal punto di vista computazionale. Generalizzando il caso appena preso in esame, possiamo costruire in pseudo-codice l'algoritmo definitivo per il sottocampionamento di un segnale digitale:

```
prima = 48000; dopo = 12800;
step = prima / dopo; p = 0;

For n = 0  to dopo-1

        nuovo_campione = segnale_vecchio[int(p)] * (1 - (p-int(p)));

        For j = int(p)+1  to int(p+step)-1

                nuovo_campione += segnale_vecchio[j];

        Next

        nuovo_campione += segnale_vecchio[int(p+step)] * ((p+step)-int(p+step));
        nuovo_segnale[n] = nuovo_campione / step;
        p += step;

Next
```

- Il sottocampionamento delle immagini digitali

Le immagini digitali possono essere trattate come il caso particolare di un segnale a due dimensioni: possiamo così estendere al caso bidimensionale l'algoritmo usato in precedenza per i segnali digitali. ll nuovo campione deve essere calcolato non più in base alla media pesata di un solo vettore di campioni, ma di un'intera matrice di campioni, come è possibile vedere dalla figura qui accanto. Dobbiamo eseguire il calcolo del nuovo campione con media pesata esattamente come avevamo fatto in precedenza per un vettore di campioni, ripetendo l'operazione per ogni riga della matrice e tenendo conto che, nel bordo superiore ed in quello inferiore, i pesi di ciascun campione dovranno essere moltiplicati per un peso che è pari rispettivamente a [int(ty+1) – ty] e [(ty+sy) – int(ty+sy)]. Alla fine, la somma totale dei pesi che abbiamo attribuito a ciascun campione dovrà essere esattamente uguale a sx*sy, cioè l'area del rettangolo preso in considerazione (i pesi infatti sono proprio le aree di ogni singolo campione). Sviluppando l'algoritmo del sottocampionamento di un'immagine digitale otteniamo il seguente pseudo-codice:




```
Dim old_width, old_height
Dim new_width, new_height

sx = old_width / new_width;
sy = old_height / new_height;

For py = 0  to new_height-1

        tx = 0;

        For px = 0  to new_width-1

'riga superiore

                j = int(ty);

                new_sample = old_image[int(tx), j] * (1 - (tx-int(tx)));
                For i = int(tx)+1  to int(tx+sx)-1
                        new_sample += old_image[i, j];
                Next
                new_sample += old_image[int(tx+sx), j] * ((tx+sx)-int(tx+sx));

                new_sample = new_sample * (1 - (ty-int(ty)));

'righe intermedie

                For j = int(ty)+1  to int(ty+sy)-1

                        new_sample = old_image[int(tx), j] * (1 - (tx-int(tx)));
                        For i = int(tx)+1  to int(tx+sx)-1
                                new_sample += old_image[i, j];
                        Next
                        new_sample += old_image[int(tx+sx), j] * ((tx+sx)-int(tx+sx));
```

```
                Next

'riga inferiore

                j = int(ty+sy);

                new_sample2 = old_image[int(tx), j] * (1 - (tx-int(tx)));
                For i = int(tx)+1  to int(tx+sx)-1
                        new_sample2 += old_image[i, j];
                Next
                new_sample2 += old_image[int(tx+sx), j] * ((tx+sx)-int(tx+sx));

                new_sample = new_sample2 * ((ty+sy)-int(ty+sy));

'scrivo il nuovo campione calcolato

                new_image[px, py] = new_sample / (sx * sy);

                tx += sx;

        Next

        ty += sy;

Next
```

Immagine originale

Ridotta con stretch

Ridotta con sub-sampling

Com'è possibile vedere da questo esempio, l'immagine ridotta con lo stretch dei campioni presenta dei difetti di aliasing che non sono presenti nell'immagine ridotta con il sottocampionamento. Il subsampling è quasi sempre usato quando si devono ridurre delle immagini troppo grandi in immagini più piccole, come gli screenshot di un videogioco da inserire nell'articolo di una rivista, senza perdere qualità, anzi, nella maggior parte dei casi le immagini ridimensionate con questa tecnica risultano visivamente più gradevoli rispetto a quelle mostrate nelle loro dimensioni originali.

Per questo motivo il sottocampionamento viene implementato in una sua variante, chiamata supercampionamento (supersampling) come tecnica di antialiasing per migliorare la qualità delle immagini generate da alcuni motori grafici, programmi di raytracing, generatori procedurali di texture e campionatori di forme geometriche anche complesse (come le clipart).