Progetto C 2019/2020

Pdf esplicativo

Col presente pdf volevamo aggiungere una nota su una funzione carina aggiuntiva, che abbiamo deciso di implementare all’interno del nostro progetto e fornire spiegazioni riguardo una scelta implementativa su di una funzione, più nel particolare sulla funzione di blending.

✓ *Spiegazione funzione aggiuntiva:*

funzione rotate:

```
ip_mat * ip_mat_rotate(ip_mat * in,unsigned int degree);
```

Ci sembrava carino aggiungere una funzione che fosse in grado di ruotare in senso orario la nostra immagine a seconda del numero di gradi inseriti (0 ,90 ,180 ,270 ,360), solitamente nei maggiori programmi di fotoritocco questa può risultare una delle funzioni più essenziali, per questo abbiamo deciso di inserirla.

Funzionamento:

Come parametri formali accetta una **ip_mat** e i **gradi** di rotazione di cui si desidera appunto ruotare l’immagine (i gradi come espresso prima possono essere: 0,90,180,270,360) e restituisce una nuova **ip_mat** ruotata della gradazione inserita. Inserendo 0 come parametro effettivo di degree la funzione restituisce l'immagine specchiata orizzontalmente. Inserendo invece tutti gli altri 4 parametri, viene restituita la nostra immagine in input ruotata (in senso orario) per i rispettivi gradi.

Esempio funzionamento:

Immagine con parametro degree 0 e concat abilitato

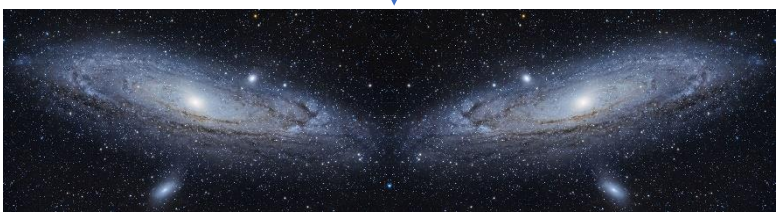

Immagine con parametro degree 90 e concat disabilitato

Note sulla funzione:

la definizione della funzione è stata inserita anche nel file **ip_lib.h** e anche la chiamata è stata inserita nel **main_iplib.c**, nel caso voleste provare il suo funzionamento. Per testarla basta lanciare da terminale la seguente dicitura:

./**main_ilplib nome_img**.bmp **nome_img**.bmp **rotate** img_finale.bmp 0 **gradi**

Come penso immaginiate non sarà possibile effettuare il concatenamento delle immagini se si effettua una rotazione di 90 o 270 gradi, in quanto le dimensioni risultati saranno invertite da quelle di partenza.

✓ *Spiegazione cambio implementazione funzione:*

Nella funzione blend inizialmente andavamo ad utilizzare le funzioni **ip_mat_mul_scalar** ed **ip_mat_sum**, andando quindi a generare rispettivamente tre matrici differenti, per ottenerne una sola in output che fosse la somma delle due generate dalle ip_mat_mul_scalar, esattamente così:

```
ip_mat * ip_mat_blend(ip_mat * a, ip_mat * b, float alpha){
    ip_mat *newa;
    ip_mat *newb;
    ip_mat *out;
    newa = NULL;
    newb = NULL;
    out = NULL;

    if (!a) {printf("Errore parametro a NULL !!!\n"); exit(1);}
    if (!b) {printf("Errore parametro a NULL !!!\n"); exit(1);}

    if ((a->h == b->h) && (a->w == b->w) && (a->k == b->k)){

        newa = ip_mat_mul_scalar(a,alpha);
        newb = ip_mat_mul_scalar(b,1-alpha);
        out = ip_mat_sum(newa,newb);

        ip_mat_free(newa);
        ip_mat_free(newb);
        return out;
    }
    printf("Errore!!! le matrici devono avere le medesime dimensioni\n");
    exit(1);
}
```

Secondo noi questo metodo andava a sfruttare più memoria del dovuto, per questo abbiamo deciso di effettuarne uno differente con più cicli rispetto a prima, eliminando però **ip_mat_mul_scalar** che generavano appunto queste due **ip_mat** non necessariamente essenziali e la **ip_mat_sum**.

La funzione risultante dopo la modifica è questa:

```
ip_mat * ip_mat_blend(ip_mat * a, ip_mat * b, float alpha){
    ip_mat *out;
    float mula,mulb;
    unsigned int v,i,j;
    out = NULL;


    if (!a) {printf("Errore parametro a NULL !!!\n"); exit(1);}
    if (!b) {printf("Errore parametro a NULL !!!\n"); exit(1);}

    if ((a->h == b->h) && (a->w == b->w) && (a->k == b->k)){

        mula = alpha;
        mulb = 1.0-alpha;

        out = ip_mat_create(a->h ,a->w ,a->k ,0.0 );

        if (out) {
            for ( v = 0; v< out->k ; v++){
                for (i = 0; i < out->h; i++){
                    for (j = 0; j < out->w; j++){

                        set_val(out ,i ,j ,v ,(get_val(a ,i ,j ,v )*mula) + (get_val(b ,i ,j ,v )*mulb) );
                    }
                }
            }
        }
        return out;
    }
    printf("Errore!!! le matrici devono avere le medesime dimensioni\n");
    exit(1);
}
```

utilizzando le funzioni **set_val** e **get_val** siamo riusciti a sfruttare meno memoria effettuando qualche ciclo aggiuntivo e mantenendo il risultato della funzione inalterato.

✓ *Feedback generale progetto:*

Volevamo approfittare per lasciare un piccolo feedback al progetto, possiamo dire di aver trovato il tutto molto interessante ed appagante, infatti una delle cose più soddisfacenti era vedere i risultati finali dei vari filtri applicati sulle immagini.
Unica nota negativa che abbiamo trovato, forse, è stata la tendenza a mantenere una linearità troppo ristretta per alcuni tratti del progetto.
Per il resto possiamo dire di esserci divertiti molto nello sviluppo del tutto. 🙂
Grazie dell’attenzione.
Gasparello Alessandro.
Simonato Alessandro.