Progetto corso programmazione 2020 - Image processing

-Scelte progettuali:

Ho creato un paio di piccole macro per rendere il codice meno ridondante:

- “**tripleFor**” che evita di scrivere ogni volta 3 for per iterare la matrice.
- “**max**”, “**min**” per calcolare il minimo/massimo di 2 valori.

```
#define min(a,b) (((a)<(b))?(a):(b))
#define max(a,b) (((a)>(b))?(a):(b))

#define tripleFor(mat,i,j,k) for(i=0;i<mat->h; i++) for( j=0; j<mat->w;j++) for( k=0;k<mat->k;k++)
```

Una piccola funzione per “scovare” gli errori che riceve come parametro una stringa (l’errore da stampare).

```
void ex(char* ex){
    printf("\n Errore %s.....", ex);
    exit(1);
}
```

Ho lasciato una vecchia versione di ***get_normal_random()*** in quanto il corrupt l’avevo fatto prima delle modifiche del progetto.

```
float get_old_normal_random(){
    float y1 = ( (float)(rand()) + 1. )/( (float)(RAND_MAX) + 1. );
    float y2 = ( (float)(rand()) + 1. )/( (float)(RAND_MAX) + 1. );
    return cos(2*PI*y2)*sqrt(-2.*log(y1));
}
ip_mat * ip_mat_corrupt(ip_mat * a, float amount){
    unsigned int i,j,k;
    ip_mat* ris=ip_mat_copy(a);
    tripleFor(a,i,j,k)set_val(ris,i,j,k,get_val(ris,i,j,k)+(amount*get_old_normal_random()));
    return ris;
}
```

Ho usato pero’ la nuova versione della funzione per fare ***l’ip_mat_init_random()***;

```
void ip_mat_init_random(ip_mat * t, float mean, float var){
    unsigned int i,j,k;
    tripleFor(t,i,j,k)set_val(t,i,j,k,(get_normal_random(mean,sqrt(var))));
}
```

```
ip_mat * to_ip_mat(float *mat,int rows,int cols,int canals) {
    ip_mat *f;
    unsigned int i,j,k;
    f = ip_mat_create(rows, cols, canals, 0.0F);
    tripleFor(f,i,j,k)
                set_val(f,i,j,k,(*((mat+i*cols) + j)));
    return f;
}
```

Ho creato una funzione che converte una matrice in una *ip_mat* per convertire i filtri (creati con una matrice appunto) in una *ip_mat*.

Un esempio:

```
ip_mat * create_sharpen_filter(){
    const float mat[3][3] = {{0,-1,0},{-1,5,-1},{0,-1,0}};
    return to_ip_mat((float *)mat,3,3,3);
}
```

Ho creato altri due metodi che modificano l'immagine:

Specular

Data un'immagine crea l'effetto specchio e la capovolge secondo l'asse delle ordinate.

```
ip_mat * spec(ip_mat * a){
    unsigned int i,j,k;
    ip_mat * mat=ip_mat_create(a->h, a->w,a->k, 0);

    for( i=0; i<a->h;i++) {
        for ( j = 0; j < a->w ; j++) {
            for( k=0;k<a->k;k++){
                set_val(mat,i,(a->w)-(1+j),k,get_val(a,i,j,k));
            }
        }
    }
    return mat;
}
```

La seconda funzione: **Minecraft**

Data un'immagine, la rende pixellata:

Come si puo' notare le 2 funzioni non usano alcun kernel/matrice come filtro.

```
ip_mat * mine(ip_mat * a,int amount){
    int jj=0;
    unsigned int i,j,k;
    ip_mat * mat=ip_mat_create(a->h, a->w,a->k, 0);

    for( i=0; i<a->h;i++) {
        for ( j = 0; j < (a->w); j++) {
            if(j%amount==0)jj=j;
            for( k=0;k<(a->k);k++){
                set_val(mat,i,j,k,get_val(a,i,jj,k));
            }
        }
    }
    return mat;
}
```

Oppure:

Attenzione: Per testare le due funzioni, inserire prima la firma del metodo nei headers!

-David Ambros