CR TP4 COMMANDRE

Benjamin COMMANDRE

27 Février 2015

1 Création d'une image couleur au format ppm

Image utilisée :

FIGURE 1 – chat.ppm

URL : `http://www.online-image-editor.com//styles/2014/images/example_image.png`

2 Création d'une image en niveau de gris

```
void couleurToGris(char* entre, char* sortie){

        int nH, nW, nTaille;

        OCTET *ImgIn, *ImgOut;

        lire_nb_lignes_colonnes_image_ppm(entre, &nH, &nW);
        nTaille = nH * nW;
```

```
    int nTaille3 = nTaille * 3;
    allocation_tableau(ImgIn, OCTET, nTaille3);
    lire_image_ppm(entre, ImgIn, nH * nW);
    allocation_tableau(ImgOut, OCTET, nTaille);

    for(int i=0; i<nTaille3; i++){
            ImgOut[i / 3] = (ImgIn[i] + ImgIn[i + 1] + ImgIn[i + 2] ) / 3;
    }

    ecrire_image_pgm(sortie, ImgOut, nH, nW);

    free(ImgIn);
    free(ImgOut);

}
```

FIGURE 2 – Image en niveau de gris

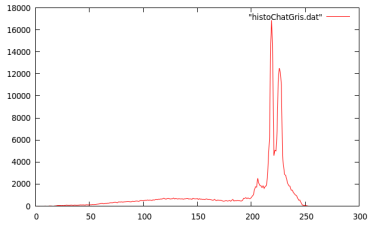

FIGURE 3 – Histogramme de l'image en niveau de gris

3 Seuillage de l'image

Algorithme pour seuiller en niveau de gris :

```
void seuilGris(char* entre, char* sortie, int seuil){
        int nH, nW, nTaille;

        OCTET *ImgIn, *ImgOut;

        lire_nb_lignes_colonnes_image_pgm(entre, &nH, &nW);
        nTaille = nH * nW;

        allocation_tableau(ImgIn, OCTET, nTaille);
        lire_image_pgm(entre, ImgIn, nH * nW);
        allocation_tableau(ImgOut, OCTET, nTaille);

        for(int i=0; i<nTaille; i++){
                if(ImgIn[i] < seuil){
                        ImgOut[i] = 0;
                }else{
                        ImgOut[i] = 255;
                }
        }

        ecrire_image_pgm(sortie, ImgOut, nH, nW);

        free(ImgIn);
        free(ImgOut);

}
```

Pour cette image, un seuil de 210 conviens.

FIGURE 4 – Image en niveau de gris seuillée

4 Floutage de l'image couleur

```
void floutageCouleur(char* entre, char* sortie){

int nH, nW, nTaille;

OCTET *ImgIn, *ImgOut;


lire_nb_lignes_colonnes_image_ppm(entre, &nH, &nW);
nTaille = nH * nW;

int nTaille3 = nTaille * 3;

allocation_tableau(ImgIn, OCTET, nTaille3);
lire_image_ppm(entre, ImgIn, nH * nW);
allocation_tableau(ImgOut, OCTET, nTaille3);

for(int i=0; i<nTaille3; i+=3){

        ImgOut[i] = ( ImgIn[i] + ImgIn[i - 3] + ImgIn[i + 3] +
        ImgIn[i - 12] + ImgIn[i + 12] + ImgIn[i + 15] + ImgIn[i - 15]
        + ImgIn[i - 18] +ImgIn[i - 18]  ) / 9;

        ImgOut[i+1] = ( ImgIn[i+1] + ImgIn[i+1 - 3] + ImgIn[i+1 + 3] +
        ImgIn[i+1 - 12] + ImgIn[i+1 + 12] + ImgIn[i+1 + 15] +
```

```
            ImgIn[i+1 - 15] + ImgIn[i+1 - 18] + ImgIn[i+1 - 18])/ 9;

            ImgOut[i+2] =  ( ImgIn[i+2] + ImgIn[i+1 - 3] + ImgIn[i+2 + 3] +
            ImgIn[i+2 - 12] + ImgIn[i+2 + 12] + ImgIn[i+2 + 15] +
            ImgIn[i+2 - 15] + ImgIn[i+2 - 18] + ImgIn[i+2 - 18])/ 9;

}

ecrire_image_ppm(sortie , ImgOut, nH, nW);

free(ImgIn);
free(ImgOut);

}
```

FIGURE 5 – Image couleur floutée

5 Floutage du fond de l'image couleur

```
void floutageFond(char* entre,char* imgGris ,char* sortie){

int nH, nW, nTaille;
int nHG, nWG, nTailleG;

OCTET *ImgIn, *ImgOut, *ImgGris;

lire_nb_lignes_colonnes_image_ppm(entre, &nH, &nW);
nTaille = nH * nW;
```

```
lire_nb_lignes_colonnes_image_pgm(imgGris, &nHG, &nWG);
nTailleG = nHG * nWG;

int nTaille3 = nTaille * 3;

allocation_tableau(ImgIn, OCTET, nTaille3);
lire_image_ppm(entre, ImgIn, nH * nW);
allocation_tableau(ImgGris, OCTET, nTailleG);
lire_image_pgm(imgGris, ImgGris, nHG * nWG);
allocation_tableau(ImgOut, OCTET, nTaille3);

for(int i=0; i<nTaille3; i+=3){

        if(ImgGris[i/3] == 255){

                ImgOut[i] = ( ImgIn[i] + ImgIn[i - 3] + ImgIn[i + 3] +
                ImgIn[i - 12] + ImgIn[i + 12] + ImgIn[i + 15] +
                ImgIn[i - 15] + ImgIn[i - 18] + ImgIn[i - 18] ) / 9;

                ImgOut[i+1] = ( ImgIn[i+1] + ImgIn[i+1 - 3] +
                ImgIn[i+1 + 3] + ImgIn[i+1 - 12] + ImgIn[i+1 + 12] +
                ImgIn[i+1 + 15] + ImgIn[i+1 - 15] + ImgIn[i+1 - 18] +
                ImgIn[i+1 - 18]  ) / 9;

                ImgOut[i+2] =  ( ImgIn[i+2] + ImgIn[i+1 - 3] +
                ImgIn[i+2 + 3] + ImgIn[i+2 - 12] + ImgIn[i+2 + 12] +
                ImgIn[i+2 + 15] + ImgIn[i+2 - 15] + ImgIn[i+2 - 18] +
                ImgIn[i+2 - 18]   ) / 9;

        }else{
                ImgOut[i] = ImgIn[i];
                ImgOut[i+1] = ImgIn[i+1];
                ImgOut[i+2] = ImgIn[i+2];

        }
}

ecrire_image_ppm(sortie, ImgOut, nH, nW);

free(ImgIn);
free(ImgOut);
}
```

FIGURE 6 – Image couleur floutée partiellement

6 Erosion et dilatation

6.1 Algorithme d'erosion

```
void erosion(char* entre, char* sortie){

        int nH, nW, nTaille,nB;

        OCTET *ImgIn, *ImgOut;

        lire_nb_lignes_colonnes_image_pgm(entre, &nH, &nW);
        nTaille = nH * nW;

        allocation_tableau(ImgIn, OCTET, nTaille);
        allocation_tableau(ImgOut, OCTET, nTaille);
        lire_image_pgm(entre, ImgIn, nH * nW);


        for (int i=0; i < nH; i++){
                for (int j=0; j < nW; j++){
                        nB = 0;
                        for(int k=-1; k<2; k++){
                                for(int l=-1; l<2; l++){
                                        if(ImgIn[(i + k)*nW+(j+l)] == 255 ){
                                                nB++;
                                        }
                                }
                        }
```

```
                        if(nB >= 4){
                                ImgOut[i*nW+j] = 255;
                        }else{
                                ImgOut[i*nW+j] = ImgIn[i*nW+j];
                        }
                }
        }

        ecrire_image_pgm(sortie, ImgOut,  nH, nW);

        free(ImgIn);
        free(ImgOut);
}
```

6.2 Algorithme de dilatation

```
void dilatation(char* entre,char* sortie){

        int nH, nW, nTaille,nB;

        OCTET *ImgIn, *ImgOut;

        lire_nb_lignes_colonnes_image_pgm(entre, &nH, &nW);
        nTaille = nH * nW;

        allocation_tableau(ImgIn, OCTET, nTaille);
        allocation_tableau(ImgOut, OCTET, nTaille);
        lire_image_pgm(entre, ImgIn, nH * nW);


        for (int i=0; i < nH; i++){
                for (int j=0; j < nW; j++){
                        nB = 0;
                        for(int k=-1; k<2; k++){
                                for(int l=-1; l<2; l++){
                                        if(ImgIn[(i + k)*nW+(j+l)] == 0 ){
                                                nB++;
                                        }
                                }
                        }

                        if(nB >= 4){
                                ImgOut[i*nW+j] = 0;
                        }else{
```

```
                                        ImgOut[i*nW+j] = ImgIn[i*nW+j];
                                }
                        }
                }

                ecrire_image_pgm(sortie, ImgOut,  nH, nW);

                free(ImgIn);
                free(ImgOut);
}
```

6.3 Combinaison

```
void ouverture(char* entre, char* sortie){
        erosion(entre,sortie);
        dilatation(sortie,sortie);
}
```

FIGURE 7 – Image binaire traitée

FIGURE 8 – Fond flouté avec la seconde image binaire