# Rapport RIM 2018

Gennuso Tàzio

# 1. Génération d’histogrammes

## 1.1. Principe des histogrammes

L’histogramme d’une image est un vecteur décrivant les quantités de chaque couleur (ou chaque niveau de gris) présent dans l’image.

Pour une image, un histogramme est une représentation condensée qui facilite son identification, son indexation et la comparaison à d’autres images, car c’est un objet bien plus léger et facile à manipuler que l’image telle qu’elle.

### 1.1.1 Histogramme en niveau de gris

L’histogramme en niveau de gris peut se voir comme un espace fini discrétisé à 1 dimension (une ligne de points à intervalles réguliers), où un poids est associé à chaque valeur (chaque point), de telle sorte que la somme de ces poids vaille 1.

La dimension (unique) représente les niveaux de gris, de blanc à noir et la discrétisation en « bins » détermine la précision de l’histogramme.

Pour chaque pixel de l’image, on détermine son intensité lumineuse (i.e. son niveau de gris) grâce à la formule suivante :

Niveau de gris = 0.299*Rouge + 0.587*Vert + 0.114*Bleu

Ce niveau de gris calculé permet de déduire la case de l’histogramme à incrémenter de 1, et finalement on normalise les valeurs de l’histogramme en divisant chaque valeur par (hauteur de l’image en pixels)*(largeur de l’image en pixels).

### 1.1.2 Histogramme en couleurs RGB

L’histogramme RGB peut se voir comme un espace fini discrétisé à 3 dimensions (un cube composé de petits cubes tous de même taille), où un poids est associé à chaque valeur (chaque petit cube), de telle sorte que la somme de ces poids vaille 1.

Les trois dimensions représentent les intensités de blanc à rouge, de blanc à vert et de blanc à bleu, et la discrétisation en « bins » détermine la précision de l’histogramme. Cette discrétisation est identique pour chaque dimension.

Pour chaque pixel de l’image, on incrémente la case de l’histogramme dont les coordonnées sont déterminées par l’intensité de chacune des couleurs rouge, vert et bleu.

Ce niveau de gris calculé permet de déduire la case de l’histogramme à incrémenter de 1, et finalement on normalise les valeurs de l’histogramme en divisant chaque valeur par (hauteur de l’image en pixels)*(largeur de l’image en pixels).

## 1.2. Code générant les histogrammes

### 1.2.1. ComputeGrayLevelHistogram

```
float[] ComputeGrayLevelHistogram(int bins) {
    /*L'histogramme est une liste de 'bins' flottants.*/
    float[] hist = new float[bins];
    /*La variable grey va contenir le calcul du niveau de gris pour chaque
pixel.*/
    double grey;

    /*On initialise la liste de l'histogramme à 0 partout.*/
    for (int i = 0 ; i < bins ; i++) hist[i] = 0;

    /*Pour chaque pixel de l'image, on calcule son niveau de gris,
    et on incrémente la case adéquate de l'histogramme. */
    for (int i = 0 ; i < this.width ; i++) {
        for (int j = 0 ; j < this.height ; j++) {
            grey = 0.299*this.pixels[i][j][RED] + 0.587*this.pixels[i][j][GREEN]
+ 0.114*this.pixels[i][j][BLUE];
            for (int k = 0 ; k < bins ; k++) {
                if ( (grey >= (float)k/bins) && (grey <= (float)(k+1)/bins) ) {
                    hist[k] += 1;
                }
            }
        }
    }

    /*On normalise l'histogramme. */
    for (int k = 0 ; k < bins ; k++) {
        hist[k] = (float)hist[k]/(this.width*this.height);
    }

    return hist;
}
```

### 1.2.2 ComputeRGBHistogram

```
float[][][] ComputeRGBHistogram(int bins) {
    /*L'histogramme est un tableau en 3 dimensions de flottants.*/
    float[][][] hist = new float[bins][bins][bins];
    /*Ces trois variables contiendront, pour chaque pixel de l'image,
    la valeur des 3 couleurs du pixel*/
    float rpix, gpix, bpix;
    /*Ces variables contiendront, pour chaque pixel, aux 3 coordonnées de la
case
    de l'histogramme à incrémenter.*/
    int rcord, gcord, bcord;
    boolean rtest, gtest, btest;
    float normalize = this.height*this.width;

    /*Pour chaque pixel de l'image, on détermine les coordonnées de la case de
    l'histogramme à incrémenter.*/
    for (int i = 0; i < this.width ; i++ ) {
        for (int j = 0; j< this.height ; j++ ) {
            rtest = false;
            gtest = false;
            btest = false;

            rcord = -1;
            gcord = -1;
            bcord = -1;

            rpix = this.pixels[i][j][RED];
            gpix = this.pixels[i][j][GREEN];
            bpix = this.pixels[i][j][BLUE];

            for (int r = 0; r < bins ; r++ ) {
                if ( (rpix >= (float)r/bins) && (rpix <= (float)(r+1)/bins) ){
                    rtest = true;
                    rcord = r;
                }
            }

            for (int g = 0; g < bins ; g++ ) {
                if ( (gpix >= (float)g/bins) && (gpix <= (float)(g+1)/bins) ){
                    gtest = true;
                    gcord = g;
                }
            }

            for (int b = 0; b < bins ; b++ ) {
                if ( (bpix >= (float)b/bins) && (bpix <= (float)(b+1)/bins) ){
                    btest = true;
                    bcord = b;
                }
            }

            if (gtest && btest && rtest){
                hist[rcord][gcord][bcord]+=1/(normalize);
            }
        }
    }
    return hist;
}
```

# 2. Recherche par similarité visuelle

## 2.1 Principe de la recherche par similarité visuelle

Le principe de cette recherche est de déterminer les images qui « ressemble le plus » à l'image de requête.

On défini ici la notion de *ressemblance entre deux images* par la distance entre les histogrammes des deux images. On cherche ainsi les images pour lesquelles cette distance est minimale.

Enfin, chaque histogramme étant représenté par un vecteur de n dimensions, la *distance* entre deux utilisée est ici la distance de type L2, à savoir :

$$\forall x, y \in \mathbb{R}^n, dist(x, y) = \sqrt{\left(\sum_{i=0}^{n} (x_i - y_i)^2\right)}$$

Ainsi selon ce principe, rechercher les k images les plus visuellement similaires à l'image de requête dont l'histogramme est noté $x$, c'est déterminer les k images $y_{i \in [\![1,k]\!]}$ de la bases pour lesquelles $dist(x, y_{i \in [\![1,k]\!]})$ est le plus petit.

À noter que tout type d'histogramme est utilisable, pourvu qu'il s'agisse du même type qui soit utilisé pour comparer l'image requête et les autres images.

## 2.2 Étude de résultats

### 2.2.1 niveaux de gris 256 bins VS RGB 6x6x6 bins

Entre la recherche avec l'histogramme en niveaux de gris à 256 bins et avec l'histogramme RGB à 6x6x6 bins, on remarque que la seconde recherche donne des résultats bien plus pertinents.

L'image requête était une photo de champignon dans les tons verts-jaunes un peu passés.
On constate que la première recherche renvoie des images dont la luminosité globale apparaît similaire à l'œil, mais certaines ont des couleurs vraiment différentes (on voit du bleu, du orange, du rouge, du blanc…). Deux ou trois images seulement (et même pas les premières proposées) ressemblent plus à l'image requête à la fois par leurs couleurs et leurs luminosités, mais sur aucune ne figure de champignon.
La seconde recherche cependant fournit des images aux couleurs bien plus similaires, dont deux sont des champignons. On y trouve également de fond marin, un animal, mais également des gens attablés et un parterre de fleurs colorées.

### 2.2.2 RGB 6x6x6 VS RGB 4x4x4 VS RGB 2x2x2

Les trois recherches renvoient des images dans des tons de couleurs similaires à l'image requête, mais de plus en plus proches avec l'augmentation de la précision de l'histogramme.

La recherche RGB 2x2x2 n'est pas très pertinente, on y trouve un seul autre champignon (qui ne figure pourtant pas dans les autres recherches), d'autres plantes mais aussi un poisson et des éléments architecturaux.

Au contraire, les résultats de la recherche RGB 4x4x4 sont peut-être meilleurs que ceux de la recherche 6x6x6 discutés à la partie précédente. En effet, même si l'on y trouve qu'un seul champignon (également présent dans la recherche RGB 6x6x6), les autres images ne montrent que des scènes de nature ressemblant à l'image requête. On a des fleurs, des coraux ou coquillages, et un animal s'abreuvant à une mare en forêt.

Il semblerait que la quantification RGB 4x4x4 fournisse les meilleurs résultats, car elle découpe l'image en zones assez flagrantes, là où la quantificatione RGB 6x6x6 est trop proche de l'image initiale, et où la quantification RGB 2x2x2 dégrade trop l'image.

## 2.3 Résultats de recherches


*Illustration 1: Recherche avec histogramme gris 256 bins*

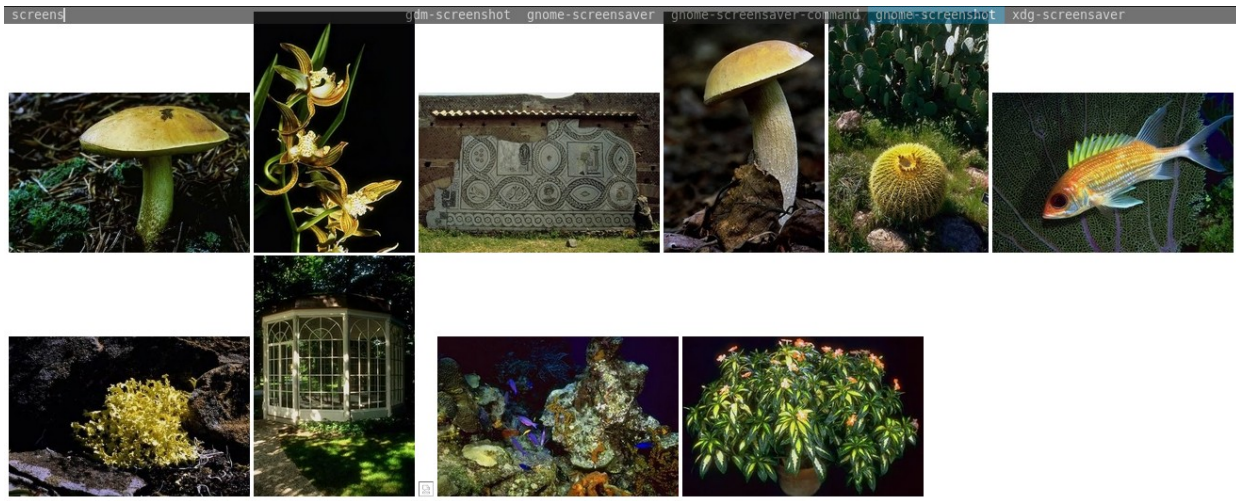


*Illustration 2: Recherche avec histogramme RGB 2x2x2 bins*

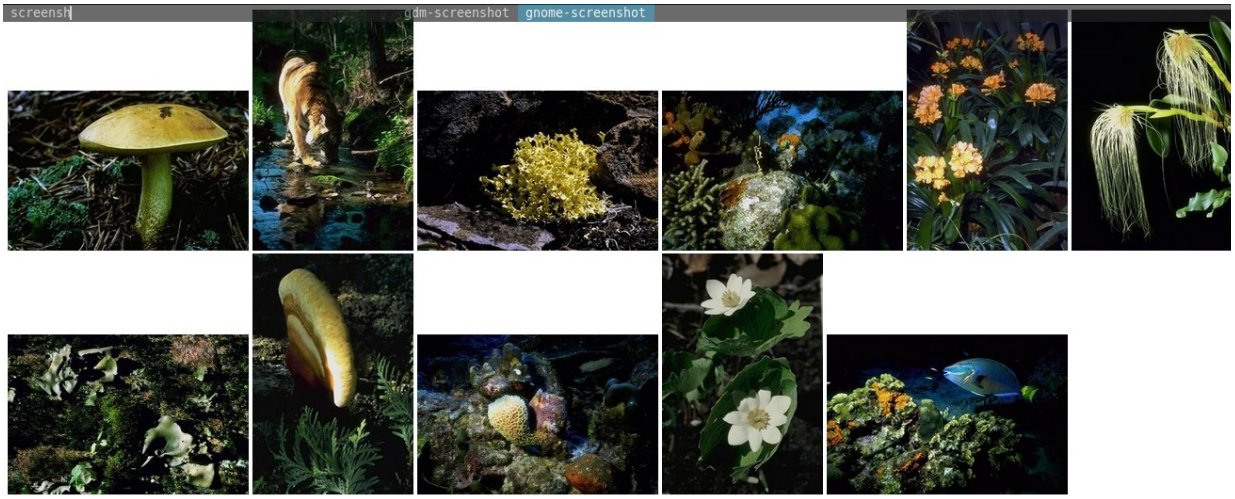


*Illustration 3: Recherche avec histogramme RGB 4x4x4 bins*

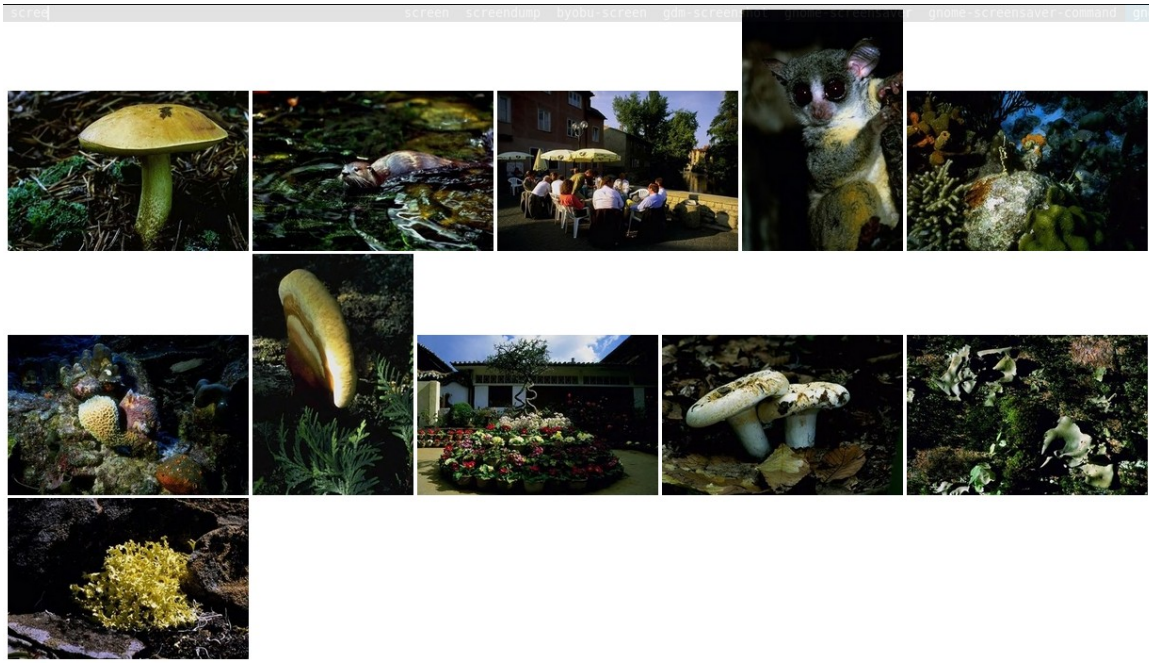


*Illustration 4: Recherche avec histogramme RGB 6x6x6 bins*