[Visu-M2IM] TP 1 : Aliasing

Vincent Feuvrier (vincent.feuvrier@normalesup.org)

1 Moirés dans les hautes fréquences

Notons Δ le carré $[-1,1]^2$ et considérons pour $\alpha > 0$ la fonction ψ_α :

$$\psi_\alpha : \left\{ \begin{array}{l} \Delta \longrightarrow \mathbb{R} \\ z = (x,y) \longmapsto \cos(2\pi\alpha|z|^2) \end{array} \right. .$$

On demande un programme générant une image en niveaux de gris de taille 200×200 correspondant à l'échantillonnage de la fonction ψ_α. Pour cela on devra procéder aux ajustements suivants :

- transformer à l'aide d'une fonction affine du plan les coordonnées entières des pixels de l'image de façon à ce que le pixel supérieur gauche corresponde au point $(-1,-1)$ de Δ et le pixel inférieur droit au point $(1,1)$;
- transformer à l'aide d'une fonction affine les valeurs de ψ_α de façon à ce que -1 corresponde à du noir et 1 à du blanc.

On testera le programme avec les valeurs suivantes de α : $\alpha = 1$, $\alpha = 10$, $\alpha = 100$ et $\alpha = 1000$. Pour quelle valeur les moirés commencent-ils à apparaitre ?

Dans un deuxième programme on définira une fonction :

```
void underSample2(Bitmap<float> *source,Bitmap<float> *dest);
```

qui sous-échantillonne l'image `source` en réduisant ses dimensions de moitié et en moyennant ses pixels par groupes de taille 2×2, et stocke l'image résultante dans `dest`. En effectuant comme précédemment l'échantillonnage de ψ_α sur une image de taille 400×400 et en la réduisant à la taille 200×200 grâce à la fonction `underSample2`, que constate-t-on pour $\alpha = 25$? Comparer avec le programme précédant.

Le fichier `bitmap.h`, téléchargeable à l'adresse `/home/feuvrier/TP/h/bitmap.h`, à inclure dans vos programmes, définit la classe `Bitmap` permettant de manipuler des fichiers Bitmap au format `RGB24`. On donne ici la liste des membres publics de cette classe :

```
template <typename T=unsigned char> struct RGB {
  T R,G,B;
}

template <typename T=unsigned char> class Bitmap {
  public:
    typedef struct RGB<T> RGB;

    Bitmap();
    Bitmap(char *filename);
    Bitmap(const int size);
    Bitmap(const int width,const int height);

    ~Bitmap();

    RGB &operator()(int x,int y);

    RGB rgb(T r,T g,T b) const;
    RGB gray(T value) const;

    RGB black()  const;
    RGB white()  const;
```

```
    RGB red()     const;
    RGB green()   const;
    RGB blue()    const;
    RGB yellow()  const;
    RGB cyan()    const;
    RGB violet()  const;

    int getWidth(void) const;
    int getHeight(void) const;

    void setSize(const int width,const int height);
    void setSize(const int size);
    void setWidth(const int width);
    void setHeight(const int height);

    void loadStream(std::istream &s);
    void saveStream(std::ostream &s) const;

    void loadFile(const char *filename);
    void saveFile(const char *filename);
};
```

2 Crênelage des lignes

Soient (x_1, y_1) et (x_2, y_2) les coordonnées de deux pixels distincts d'une image, définissant une droite de pente α. Pour simplifier on supposera que :

$$\alpha \in [-1, 1] \quad \text{et} \quad x_2 > x_1 \ .$$

L'algorithme de tracé de segment de Bresenham est le suivant :

pour x variant de x_1 à x_2, allumer le pixel de coordonnées $\left(x, y_1 + \texttt{round}\left(\dfrac{(x - x_1)(y_2 - y_1)}{x_2 - x_1}\right)\right)$.

Il consiste simplement à parcourir horizontalement l'image de x_1 à x_2 et à allumer pour chaque colonne le pixel dont le centre est le plus proche du segment.

On demande d'implémenter cet algorithme dans un programme, qui par exemple trace le segments d'extrémités $(5, 10)$ et $(45, 30)$ sur une image de taille 50×50. Que constate-t-on lorsqu'on observe l'image avec un fort grossissement ?

Modifier le programme de la façon suivante : pour chaque colonne de pixels (pour $x \in [x_1, x_2]$) calculer la distance $\delta(x, y)$ entre chaque pixel de la colonne et la droite :

$$\delta(x, y) = \frac{1}{\sqrt{1 + \alpha^2}} \left| y_1 + \frac{(x - x_1)(y_2 - y_1)}{x_2 - x_1} - y \right| \ .$$

Lorsque $\delta \leq 1$, attribuer une intensité égale à $255 * (1 - \delta(x, y))$ au pixel de coordonnées (x, y). Que constate-t-on cette fois par rapport à l'algorithme précédent ? Comment interpréter la pondération effectuée en terme de filtrage spatial ?

Donner en fonction des coordonnées des extrémités des segments la complexité des deux algorithmes. Comparer avec la méthode de sur-échantillonnage vue dans la première partie.