Projet Maths pour l'info

—IMAC 1—

Détection de lignes

La détection de lignes est un sujet encore d'actualité dans le domaine de la recherche en traitement d'images et vision par ordinateur. Le problème est de détecter des lignes sur une image. Ces lignes font en général plus d'un pixel de large, ne sont pas forcément completement droites et sont parfois coupées par d'autres objets de l'image. La détection de lignes est une première étape pour la reconnaissance de formes et d'objets. Elle peut aussi servir à faciliter l'utilisation de certains logiciels.

1 Position du problème

Une image (du point de vue informatique) peut être vue comme une succession de pixels (picture element) qui, disposés les uns à côté des autres, représentent une forme, une personne, ce que vous voulez... La vision par ordinateur et le traitement d'images sont deux domaines qui tentent d'analyser ces images et d'en tirer des informations. Le premier constat effectué est que ce qui parait évident pour l'homme l'est nettement moins pour la machine. Certaines tentatives ont été réalisées dans le domaine de l'intelligence artificielle mais les résultats obtenus ne sont probants que pour des situations très spécifiques, la voie la plus prometteuse reste donc l'analyse d'image.
Le premier objectif a été de détecter les contours des objets, ensuite de détecter des objets simples comme des droites ou des cercles puis des objets plus complexes commme des coniques et enfin des objets quelconques comme un visage, une voiture, une merguez ...
Dans le cadre de ce projet, nous nous limiterons aux droites et aux segments de droites. Celles-ci peuvent être de n'importe quelle taille et de n'importe quelle épaisseur, pourvu qu'elles soient composées d'un ensemble de points à peu près alignés.

2 Méthodes de résolution

La détection de droite va s'effectuer en deux parties :

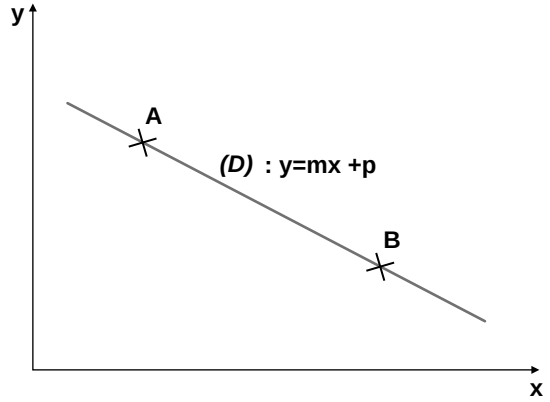
- La transformation de Hough permet de détecter une droite dans une image. Les paramètres des droites à détecter ne sont maleureusement pas toujours très précis et plusieurs résultats représentants la même droite sont souvents trouvés.
- Suppression des doublons et choix des meilleures droites par une technique de votre choix.

3 Transformation de Hough

3.1 Principe

Soit (D) une droite notée $y = mx + p$ passant par deux points distincts A et B. La droite (D) vérifie donc les équations :

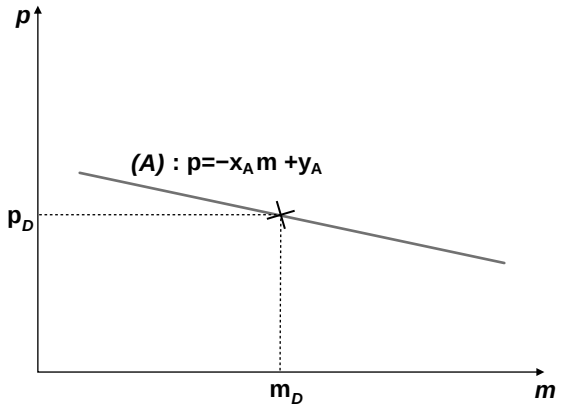
$$y_A = mx_A + p \tag{1}$$
$$y_B = mx_B + p \tag{2}$$



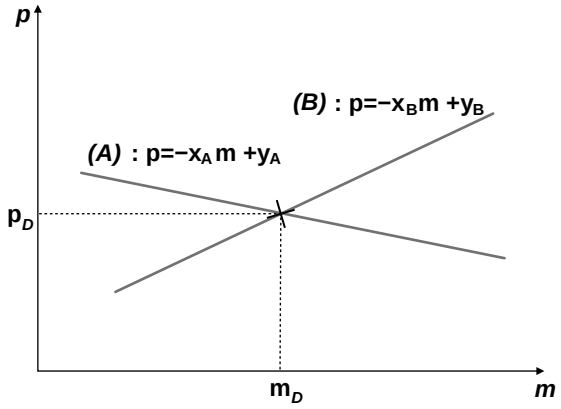
La transformation de Hough propose de changer d'espace de représentation de ces droites et de ces points de la manière suivante :

$$(A) : p = -x_A m + y_A \tag{3}$$
$$(B) : p = -x_B m + y_B \tag{4}$$

L'équation (1) engendre la droite (A) (équation(3)) de coefficient directeur $-x_A$ et d'ordonnée à l'origine y_A. Cette droite est associée spécifiquement au point A. La transformation de Hough d'un point donne donc une droite. Cette droite passe par le point (m_D,p_D) où m_D est le coefficient directeur de la droite (D) et p_D l'ordonnée à l'origine. La droite (A) passe par une infinité de couples (m_i, p_i) de points. Ces couples de points de l'espace (m, p) correspondent à toutes les droites passant par le point A dans l'espace (x, y).



En faisant de même avec le point B, on obtient la droite (B) qui passe elle aussi par le point (m_D,p_D). Ainsi, n points alignés dans l'espace (x, y) génèrent n droites sécantes dans l'espace (m, p).



Concrètement, la transformée de Hough est stockée dans "un buffer d'accumulation" c'est-à-dire un tableau à deux dimensions dans le cas de l'espace (m, p) sur lequel on dessine les droites associées aux points de l'espace (x, y). A chaque case de ce tableau on attribue un score variant selon le nombre de fois qu'elle a été parcourue par une droite. Ceci implique qu'un point intersecté par deux droites a un score supérieur à un point parcouru par une seule droite.
Pour trouver les droites présentes dans l'espace (x, y), il suffit de chercher les pixels de score maximal dans l'espace (m, p). Les pixels (m_i, p_i) choisis nous indiquent qu'un grand nombre de points sont alignés sur la droite $y = m_i x + p_i$ dans l'espace (x, y).

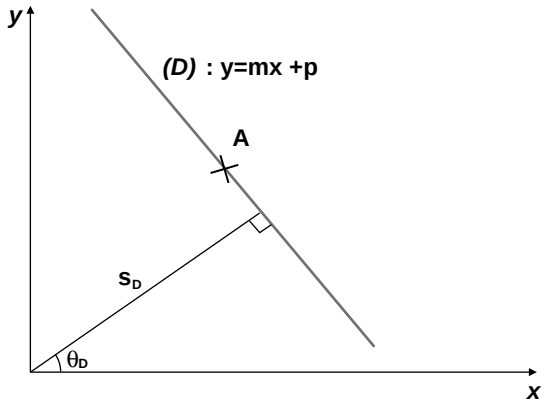
3.2 Taille du buffer et représentation polaire

En utilisant la méthode décrite précédement, on s'aperçoit rapidement que pour la plupart des points de l'espace (x, y), les coefficients directeurs des droites obtenue sont très élevés et que pour certains points, l'ordonnée à l'origine est telle que la droite n'apparait pas forcément dans le tableau (de taille finie) représentant l'espace (m, p). De tels problèmes deviennent gênant pour les droite verticales. Il peut paraître suffisant d'agrandir la taille du "buffer d'accumulation" et de le décentrer mais en réalité, ce n'est pas suffisant. En représentant une ligne le la façon suivante, ce genre de problèmes disparait :

$$s = x.cos\theta + y.sin\theta \tag{5}$$

$$\text{avec } \theta \in [0, 2\pi[$$

L'ensemble des couples (θ_i, s_i) vérifiant l'équation $s = x_A.cos\theta + y_A.sin\theta$ correspond à l'ensemble des droites passant par A en représentation polaire.



De même, tous les couples (x_i, y_i) vérifiant l'équation $s_D = x.cos\theta_D + y.sin\theta_D$ appartiennent à la droite (D).

Deux remarques :

- Il existe une valeur S telle que si $|s| > S$, la droite en question ne figure pas sur l'image originale. Autrement dit une des dimensions du "buffer d'accumulation" est fixée.
- L'angle θ varie de 0 à 2π, toutefois sur cet intervalle, la même droite est représentée deux fois : une première fois avec (θ, s) et une seconde fois avec $(\theta + \pi, -s)$. Il est donc suffisant de traiter θ sur l'intervalle $[0, \pi[$. La seconde dimension du "buffer" est donc elle aussi fixée.

Ce qui vient d'être fixé est l'échelle du "buffer d'accumulation" et non sa taille en nombre de cases. Celle-ci doit être choisie judicieusement car si le nombre de cases est trop faible, les équations de droites trouvées seront imprécises et si le nombre de cases est trop élevé, la détection deviendra longue et un nombre important de droites seront proposées pour représenter la même droite dans l'image originale.

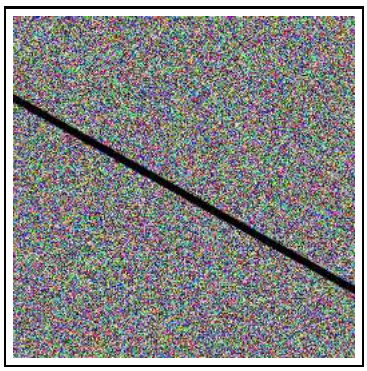
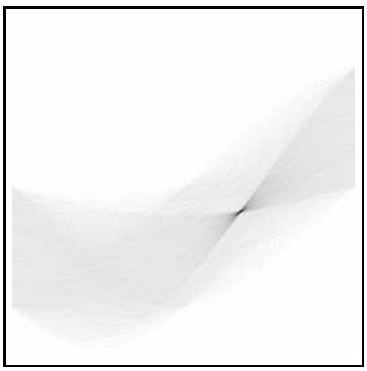
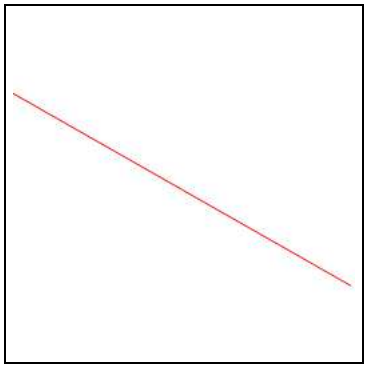


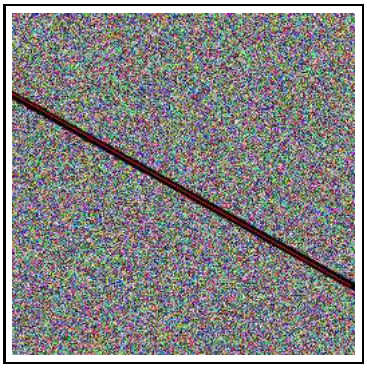
Image originale



Transformation de Hough



Droite détectée



Droite sur l'image originale

4 Suppression des doublons et choix de la meilleur droite

La transformation de Hough permet de détecter la présence de droites dans une image mais parfois propose plusieurs droites pour représenter en fait la même droite. Prenons par exemple le cas d'une ligne épaisse de quelques pixels, sa transformation de Hough proposera plusieurs lignes de un pixel de large les unes à coté des autres, or seulement une droite est représentative de cette ligne.
Ce problème vient du fait que les pixels représentant une droite sont rarement alignés et que si la droite en question fait plusieurs pixels de large, elle correspond effectivement à plusieurs droites. Ceci se traduit par un point d'intersection des courbes étalé sur plusieurs cases du "buffer d'accumulation". La case de score maximal n'est pas forcément la plus représentatrive de la droite et un calcul de barycentre sub-pixel peut par exemple donner un résulat plus précis.

5 Travail demandé

Le but du projet est d'implémenter en langage `C` un programme permettant de détecter des droites et des portions de droites sur une image (au format ppm). Votre programme sera composé de fichiers `.c` et `.h`. Il se compilera avec un `makefile` et comportera une option `-h` pour l'aide. Toutes les options seront spécifiées sur la ligne de commande et nulle part ailleurs.

Ce programme sera séparé en deux parties : la première efféctuera une détection de droites avec la transformation de Hough et générera une réplique de cette image avec les droites détectées en surimpression. La seconde partie consistera à choisir les droites les plus représentatives de l'image en utilisant une méthode de votre choix. Vous devrez justifier le choix de vos méthodes dans un rapport d'eviron 5-10 pages. Dans ce rapport figureront également les observations et les difficultés rencontrées.

6 Documentation technique

6.1 fichier*.ppm*

Il s'agit d'un format d'images reconnu par divers éditeurs d'images comme *gimp* ou *ee* (electric eyes). Il en existe plusieurs variantes : couleurs/niveaux de gris, ASCII/binaire. Le format le mieux adapté à notre problème est le format couleur en binaire. Le fichier commencera donc avec un en-tête correspondant.

Un pixel en couleur est défini par 3 nombres : un pour "la quantité" de rouge (R), un pour "la quantité" de vert (V) et un pour "la quantité" de bleu (B), noté RVB (RGB en anglais). Nous coderons ces quantités par des `unsigned char`. Un `unsigned char` est codé sur 8 bits, soit un octet. Ces nombres vont donc de 0 à 255.

Revenons au format *ppm*. Le premier nombre correspond au "numéro de variante" choisi, dans notre cas *P*6. Sur la ligne suivante, les deux nombres correspondent à la *largeur* et la *hauteur* de l'image. Enfin, le dernier nombre correspont à la valeur maximale que l'on puisse rencontrer (ici 255). Des commentaires peuvent être insérés dans l'en-tête, une ligne de commentaires commence par un #.
Chaque pixel sera écrit dans le fichier en procédant ligne par ligne, de gauche à doite, en commençant par le haut. Les pixels seront écrits sous forme de 3 octets écrits les uns à la suite des autres (RVB).

exemple:

```
#commentaires eventuels
P6
128 256
255
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y^@y
^@y^@y^@y ...
```

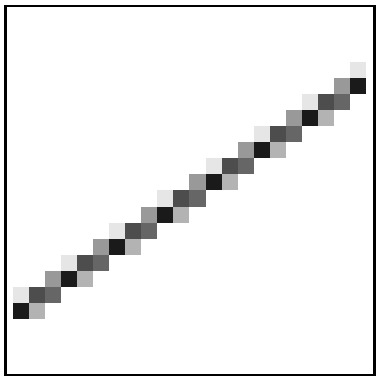
6.2 Tracer une droite avec la méthode des virgules fixes

Ce projet ne nécessite aucune bibliothèque graphique, il faut cependant pourvoir tracer une droite sur une image *.ppm*. Nous utiliserons la méthode des virgules fixes permettant de tracer de telles droites. Soit $(D) : y = m.x + p$ la droite à tracer, la première étape, appelée *clipping*, sert à délimiter parmi toute la droite la partie visible sur l'image. On ne trace en effet qu'un segment $[A, B]$ de cette droite dont il faut déterminer les coordonnées. Une fois les points A et B déterminés, il faut tracer le segment $[A, B]$. Soit $p_{i,j}$ le pixel situé sur la colonne j de la ligne i, une première approche serait de parcourir les colonnes de $j = x_A$ à x_B et de dessiner le pixel $p_{m.j+p,j}$. Malheureusement certains segments sont très mals dessinés avec cette méthode, notament les segments presque verticaux qui ne sont alors représentés que par deux ou trois pixels. Il y a donc deux cas à différencier :

- Soit $|m| \leq 1$ dans quel cas la méthode précédente est satisfaisante.
- Soit $|m| > 1$ et il est alors préférable de parcourir les ligne de $i = y_A$ à y_B et de dessiner les pixels $p_{i,\frac{i-p}{m}}$.

Le nom de la méthode proposée (méthode des virgules fixes) vient de la façon de stocker les données : prenons par exemple un `int` codé sur quatres octets, on utilise les deux octets de poids forts pour stocker les unités et les deux octets de poids faibles pour les décimales. Ainsi, les opérations de bases (addition, soustraction, multiplication et division) ne sont pas affectées.

Il est très fortement recommandé de gérer l'anti-aliasing c'est à dire que si le point à colorier ne tombe pas exactement au milieu d'un pixel, il faut "partager" la couleur avec les pixels voisins.



7 options

Il est conseillé de rajouter à votre programme quelques options. En voici quelques unes :

- gestion des couleurs.
- segmentation de la droite en plusieurs segments si la droite n'est pas affichée completement sur l'image.
- détecter des cercles ...

Vous pouvez laisser libre cours à votre imagination, toute amélioration sera la bienvenue.

vous trouverez quelques informations complémentaires à l'adresse :

`http://www-igm.univ-mlv.fr/~vnozick/`

Bon courage.