Partie 1 : Pour la transformée de Hough en espace paramétrique (m,p) :

2 Méthodes membres :

***Tracer MP :***

La méthode **tracerMP** trace une droite **y = mx + p** dans une image et enregistre cette image dans un fichier.

La méthode **tracerMP** pour la classe Droite.

Cette méthode prend en entrée les paramètres suivants :

* m et p : des constantes décrivant l'équation d'une droite y = mx + p.
* hauteur et largeur : des entiers définissant les dimensions de l'image à générer.
* maxColor : un entier définissant la valeur maximale de chaque canal de couleur RGB pour les pixels de l'image (par exemple, 255 pour une image en couleurs 24 bits).
* FILENAME : une chaîne de caractères contenant le nom du fichier de sortie.

La méthode **tracerMP** crée un objet de type ofstream pour écrire dans le fichier FILENAME en mode écriture. Elle écrit ensuite l'entête du fichier PPM dans le fichier, qui contient des informations sur les dimensions de l'image et la valeur maximale des canaux de couleur.

Ensuite, la méthode parcourt chaque pixel de l'image et calcule la valeur de y pour la droite en fonction de la valeur de x. Elle calcule également la distance entre le pixel et la droite en utilisant la formule de la distance d'un point à une droite. Cette distance permet de déterminer si le point est proche de la droite ou non, ce qui permet de décider quelle couleur lui attribuer.

Si le point est proche de la droite, la méthode colore le pixel en rouge. Si le point est légèrement éloigné de la droite, elle partage la couleur du pixel avec ses voisins pour un effet d'anti-aliasing. Si le point est loin de la droite, la méthode colore le pixel en blanc.

Notons que l’ordonnée à l’origine sera située au coin en haut à gauche de l’image. L’axe des ordonnées va vers la droite et l’axe des abscisses vers le bas.

***transformeeHoughMP :***

On lit notre image au format PPM depuis un fichier, la méthode parcourt chaque pixel de l'image pour détecter les pixels appartenant à une droite y=mx+p. Pour chaque point considérer comme appartenant à la droite, on accumule un certain score, dans un buffer d'accumulation, pour les cases du buffer, suffisamment proche d’une droite. Cette droite de l’espace (m,p), correspond à la transformée de Hough, du point repérer comme appartenant à la droite, dans l’image, soit l’espace (x,y).

Le résultat est un vecteur de valeurs représentant le nombre d'occurrences de chaque combinaison (m,p) de la droite dans le buffer d'accumulation.

Détail du code :

* Ouvre le fichier image.ppm contenant l'image.
* Lit les informations relatives à l'image depuis l'entête du fichier PPM (format, dimensions, nombre de couleurs).
* Crée des vecteurs disc\_M et disc\_P représentant les valeurs discrètes de m et p pour la discrétisation de l'espace Hough.
* Initialise un buffer d'accumulation de la transformée de Hough avec des zéros pour chaque cellule.
* Parcours chaque pixel de l'image.
* Pour chaque pixel, vérifie si la couleur du pixel correspond à celle de la droite y=mx+p (r>192, b=0, g=0).
* Si la couleur détectée est suffisamment rouge(r>192), appelle une fonction **incrementeMP** qui incrémente les cellules appropriées du buffer d'accumulation.
* Ferme le fichier.
* Récupère le score maximal obtenu dans le buffer.
* Récupère les coordonnées de toutes les cases ayant obtenues ce score maximal.
* Appelle la fonction **trouve\_barycentre** afin d’obtenir les coordonnées du barycentre sub-pixel du buffer.
* Affiche le résultat de la méthode et le buffer accumulé pour vérification des résultats et du bon fonctionnement de la méthode.
* Retourne le vecteur des valeurs de discrétisation des paramètres (m,p) pour les coordonnées correspondants à cette case (i ème ligne donne i ème coefficient de disc\_M et j ème colonne donne le j ème coefficient de disc\_P)

1 méthode spécifique :

***IncrementeMP :***

Notre méthode incrémente prend en entrée un buffer, les coordonnées d'un pixel (x,y), ainsi que des informations sur la largeur, la hauteur et la discrétisation de la droite (disc\_M et disc\_P).

Pour chaque ligne du buffer, notre fonction calcule la distance entre chaque case du buffer et la droite définie par le pixel (x,y), de coefficient directeur -x et d’ordonnée à l’origine y, et incrémente le buffer aux indices correspondants en fonction de cette distance. Les incréments sont effectués de manière graduée, avec une incrémentation plus importante pour les pixels les plus proches de la droite.

La fonction retourne le buffer mis à jour.

Détail du code :

La fonction parcourt chaque élément du buffer en utilisant deux boucles for imbriquées pour parcourir les indices i et j. Pour chaque élément, elle calcule la distance d entre le pixel (i,j) et la droite p= -mx + y. Si cette distance est inférieure à 0.5, la case correspondante du buffer est incrémentée de 1. Si la distance est comprise entre 0.5 et 1.0, l'incrément est de 0.5, et si la distance est comprise entre 1.0 et 1.5, l'incrément est de 0.1. Les pixels dont la distance est supérieure à 1.5 ne contribuent pas à l'accumulation.

Partie 2 : Pour la transformée de Hough en espace paramétrique (r,théta) :

2 Méthodes membres :

***TracerRTheta :***

La méthode crée une image en format PPM représentant une droite dans un plan cartésien. La droite est représentée en coordonnées polaires avec les paramètres r et theta.

La fonction prend en entrée les dimensions de l'image (hauteur et largeur), la valeur maximale de couleur (maxColor), le nom du fichier de sortie (FILENAME) et les paramètres polaires de la droite (r et theta).

Elle suit le même mode de fonctionnement que tracerRTheta, on change seulement l’équation de la droite, qui est maintenant en coordonnées polaires, pour la calcul de la distance.

Détail du code :

Le parcours des pixels de l'image est effectué dans une boucle imbriquée for, et pour chaque pixel, on calcule sa projection sur la droite représentée par r et theta en coordonnées polaires. La distance entre ce point et la droite est ensuite calculée et utilisée pour déterminer la couleur à attribuer au pixel.

Si cette distance est inférieure ou égale à 0.5, le pixel est colorié en rouge. Si la distance est comprise entre 0.5 et 1, la couleur rouge est atténuée en fonction de la distance. Les pixels dont la distance est supérieure à 1 sont coloriés en blanc.

Le fichier image de sortie est créé en écrivant l'entête du fichier PPM et en écrivant les valeurs de couleur de chaque pixel de l'image dans le fichier.

***TransformeeHoughRTheta :***

La méthode implémente la transformation de Hough pour détecter les droites présentes dans une image.

Plus précisément, il effectue la transformée de Hough dans l'espace paramétrique (r,theta) pour détecter les droites qui passent par des points blancs de l'image.

Détail du code :

* Le code lit une image au format .ppm et récupère les dimensions de l'image à partir du fichier. Il utilise ensuite deux tableaux disc\_R et disc\_Theta pour discrétiser l'espace paramétrique r-theta. Le tableau disc\_R contient les valeurs discrètes de r, et disc\_Theta contient les valeurs discrètes de thêta.
* Il parcourt ensuite tous les pixels de l'image et incrémente le buffer d'accumulation pour chaque pixel considéré comme suffisamment rouge.
* La fonction **incrementeRTheta** est appelée pour chaque pixel blanc, et cette fonction incrémente les cases correspondantes du tableau d'accumulation en utilisant les valeurs de r et de theta calculées pour le pixel courant.
* Parcourt le tableau d'accumulation pour trouver les coordonnées des cases ayant la valeur maximale. Ces coordonnées correspondent aux paramètres r et theta des droites détectées par la transformée de Hough, dans les vecteurs disc\_R et disc\_Theta.
* Le résultat est stocké dans un vecteur de vecteurs de doubles resultat, qui contient les coordonnées (i, j) de toutes les cases ayant le score maximal. Chaque élément du vecteur "resultat" correspond aux coordonnées r-theta d'une droite détectée.
* On calcule le barycentre de tous ces points et on affiche les coordonnées du point trouvé, ainsi que les valeurs en (r,theta) correspondantes et les valeurs en (m,p) correspondantes.
* On affiche le buffer accumulé, pour vérifier les résultats, ainsi que le bon fonctionnement des méthodes.

3 méthodes spécifiques :

***IncrementeRTheta :***

La méthode incrémente prend en entrée un buffer (un tableau à deux dimensions), les coordonnées (x,y) d'un pixel, des paramètres de discrétisation (n\_valR, n\_valTheta, disc\_R, disc\_Theta) et la taille de la droite (R).

Elle calcule la distance entre la droite définit par le pixel (x,y) et tous les couples (r,theta) de paramètres possibles, par leur discrétisation. Si la distance est inférieure à une certaine valeur dépendante de R et de la discrétisation, alors la case correspondante du buffer est incrémentée, d’une certaine valeur dépendant du degré du de proximité, par rapport à la droite.

Cette méthode est utilisée pour incrémenter le buffer avec les droites correspondant aux pixels, repéré comme appartenant à la droite dans l’image .ppm, en utilisant la transformée de Hough.

***Vers\_MP :***

La méthode prend deux doubles en entrée, r et theta, qui représentent respectivement la distance entre l'origine et la droite dans le système de coordonnées polaires, et l'angle entre la droite et l'axe des x. La fonction retourne un vecteur de deux doubles, m et p, qui représentent respectivement la pente et l'ordonnée à l'origine de la droite dans le système de coordonnées cartésiennes.

Pour obtenir m et p, la méhode utilise les relations suivantes :

la pente m est égale à : -cos(theta) / sin(theta).

l'ordonnée à l'origine p est égale à : r / sin(theta).

Ces formules sont déduites de l'équation de la droite y = mx + p dans le système de coordonnées cartésiennes, et de la conversion entre les coordonnées polaires et cartésiennes.

***Vers\_polaire :***

La méthode prend en entrée deux nombres réels m et p correspondant respectivement aux coefficients directeurs et à l'ordonnée à l'origine d'une droite de l'espace cartésien et retourne un vecteur contenant deux nombres réels correspondant aux coordonnées polaires de cette droite, à savoir la distance r de la droite à l'origine et l'angle theta entre la droite et l'axe des abscisses.

Détail du code :

La méthode calcule d'abord la distance r à partir de m et p en utilisant la formule r = abs(p) / sqrt(1 + m \* m), puis calcule l'angle theta à partir de m, et enfin retourne un vecteur contenant r et theta.

***Affiche\_buffer :***

Cette fonction affiche le buffer donnée en paramètre, coefficient par coefficient, à l’aide de double for, itérant chacune sur une de ses dimensions.

Partie 3 : Pour trouver la meilleure approximation, s’il y a plusieurs cases du buffer, qui sont de score maximal

***Trouve\_barycentre :***

Notre fonction prend en entrée un tableau de deux dimensions d'entiers, où chaque sous-tableau contient deux éléments représentant les coordonnées x et y d'un point. Il calcule ensuite les coordonnées du barycentre des points du tableau.

Le barycentre est calculé en effectuant une moyenne pondérée des coordonnées x et y des points. Pour chaque coordonnée, la moyenne pondérée est calculée en divisant la somme des coordonnées de tous les points par le nombre total de points dans le tableau. On choisit de mettre le même poids, pour chacun des points, on fait donc la moyenne des coordonnées de tous les points de la liste.

Le résultat est retourné sous forme d'un vecteur contenant les coordonnées x et y du barycentre.