**Détection de feuille**

**Principe :**

A partir de la photo qui est prise, nous allons rééchantillonner l’image afin de diminuer le nombre de pixels présents sur l’image (pour que la détection se fasse plus rapidement). A partir de cette image rééchantillonnée, nous allons appliquer un seuillage afin de différencier la feuille du support. Ce seuillage va permettre d’obtenir un tas de pixels au niveau des rebords de la feuille, mais aussi ailleurs car il y aura toujours un peu de bruit.

Afin d’obtenir les 4 droites propres à la feuille, nous allons nous servir de l’agorithme de RANSAC. A l’aide de ces 4 droites, nous en déduisons les 4 coins de la feuile, qui sont les 4 points caractéristiques nécessaires à la localisation de la feuille.

**Seuillage avec calcul de gradient :**

Afin d’utiliser cette méthode, il faut transformer l’image en niveaux de gris. Nous allons utiliser la luminance. Voici le pseudo-code correspondant :

Entrées :

Image rééchantillonnée

Sorties :

Image en niveau de gris

Algorithme :

**Pour** chaque pixel P(R,V,B) de l’image *(avec R = rouge, V = vert, B = bleu)*

Luminance(P) := 0,2 × *R* + 0,7 × *V* + 0,1 × *B*

Une image en niveau de gris est un champ de scalaire. Le gradient d'une image est un champ de vecteurs qui appartiennent au plan de l'image et se décomposent sur le repère de l'image. Nous parlons donc gradient horizontal et de gradient vertical.

Par conséquent, afin d’extraire les lignes verticales, nous allons utiliser le gradient horizontal, et vice-versa.

Soit P(x,y) un point de l’image, on calcule son gradient horizontal : .   
Si on considère que P est un point de fort gradient horizontal et est donc potentiellement un point de l’un rebord vertical.

De même pour les rebords horizontaux, si , P est un point de fort gradient vertical.

**Pseudo-code du seuillage :**

Entrées :

Image en niveau de gris

Sorties :

Image seuillée

Algorithme :

pointsRebordsVerticaux := ø

pointsRebordsHorizontaux := ø

**Pour** chaque pixel P de l’image

gradHorizontal := (x,y)

**si** (x,y) > s

ajouter P à pointsRebordsVerticaux

**si** (x,y) > s

ajouter P à pointsRebordsHorizontaux

**Fonctionnement de l’algorithme de RANSAC :**

Des points sont choisis au hasard et une droite est tracée. A partir de cette droite, on regarde les points à côté de cette droite, si la distance est suffisamment petite, on l’ajoute à l’ensemble de points. Quand tous les points ont été examiné, si le nombre de points dans l’ensemble de points est suffisamment élevé, on réajuste la droite et on mesure l’erreur, c’est-à-dire la somme des distance entre les points et la droite.

On choisit le jeu de donnée avec l’erreur la plus faible.

Limites et avantages de l’algorithme :

Un avantage de RANSAC est sa capacité à calculer de manière robuste les paramètres du modèle, c'est-à-dire qu'il peut estimer les paramètres avec un degré élevé de précision, même si une quantité importante de valeurs aberrantes (outliers) est présente dans les données. Un inconvénient de RANSAC est qu'il n'y a pas de limite supérieure sur le temps de calcul de ces paramètres. Lorsqu'une limite est utilisée (un nombre maximal d'itérations), la solution obtenue peut ne pas être la solution optimale. Un autre inconvénient de RANSAC est qu'elle suppose de fixer des seuils spécifiques au problème traité.

**Pseudo-code de l’algorithme de RANSAC :**

Entrées :

donnees : image seuillée

modeleDroite : une droite qui peut être ajusté à des données

echantillonMinimum : le nombre minimum de données nécessaires pour ajuster la droite

iterationsMax : le nombre maximal d'itérations de l'algorithme

seuilAppartenance : une valeur seuil pour déterminer si une donnée correspond à un   
 modèle

nombrePointsMin : le nombre de données proches des valeurs nécessaires pour faire   
 valoir que le modèle correspond bien aux données

Sorties :

meilleurModele : les paramètres du modèle qui correspondent le mieux aux   
 données (ou null si aucun bon modèle a été trouvé)

meilleurEnsemblePoints : données à partir desquelles ce modèle a été estimé

meilleureErreur : l'erreur de ce modèle par rapport aux données

Algorithme :

iterateur := 0

meilleur\_modèle := aucun

meilleur\_ensemble\_points := aucun

meilleure\_erreur := infini

**tant que iterateur < iterationsMax**

pointsAleatoires := echantillonMinimum valeurs choisies au hasard à partir des données

modelePossible := paramètres du modèle correspondant aux pointsAléatoires

ensemblePoints := pointsAléatoires

**Pour** chaque point des données pas dans pointsAleatoires

**si** le point s'ajuste au modelePossible avec une erreur inférieure à seuilAppartenance

Ajouter un point à ensemblePoints

**si** le nombre d'éléments dans ensemblePoints est > nombrePointsMin

*(ce qui implique que nous avons peut-être trouvé un bon modèle,*

*on teste maintenant dans quelle mesure il est correct)*

modelePossible := paramètres du modèle réajusté à tous les points de ensemblePoints

erreur := une mesure de la manière dont ces points correspondent au modelePossible

**si** erreur < meilleureErreur

*(nous avons trouvé un modèle qui est mieux que tous les précédents,*

*le garder jusqu'à ce qu'un meilleur soit trouvé)*

meilleurModele := modelePossible

meilleurEnsemblePoints := ensemblePoints

meilleureErreur := erreur

iterateur := iterateur +1

**retourne** meilleurModele, meilleurEnsemblePoints, meilleureErreur.