TIPE: Sécurité routière – Reconnaissance de panneaux routiers avec la transformée de Hough

Aurélien Mazaheri

Lycée Jean Baptiste Corot, Savigny s/Orge

March 31, 2021

Sommaire

I Théorie sur les filtres
- Filtres
- Pré traitement

II Théorie sur la transformée de Hough
- Théorie de la transformée de Hough
- Application de la transformée

III Algorithme
- Application de l'algorithme
- Pratique en conditions réelles

IV Comparaison et annexe
- Comparaison des résultats entre transformée de python (non seuillée) et l'algorithme (seuillage)
- Optimisation des résultats en conditions réelles

Les voitures autonomes sont pourvues de capteurs et de systèmes à la conduite dont le LIDAR. 4 capteurs à citer cf le document.

Mais ces capteurs utilisent un filtrage à seuillage fixe, la détection de formes, pourtant cruciale, est parfois grossière.

Le but est de reconnaître des panneaux dans le cadre de systèmes à la conduite. On va chercher pour cela des formes simples (droites, cercles), un polygône étant composé de segments.

Technique brevetée en 1962 par IBM. L'enjeu est relativement récent et Sobel à été l'un des premiers à s'intéresser au filtrage en 1953 (?)

Alors que certaines techniques commencent par suivre des points de contour afin de les relier par diverses méthodes, Hough propose d'accumuler des évidences sur la présence d'une forme définie au préalable. Cette méthode va se baser sur une bijection entre l'espace de l'image et l'espace des paramètres de la forme cherchée.

Pour détecter une forme sur une image, il faut d'abord mettre en évidence les reliefs les plus importants, c'est à dire les contours, ce qui correspond au gradient de l'intensité lumineuse en 2 dimensions

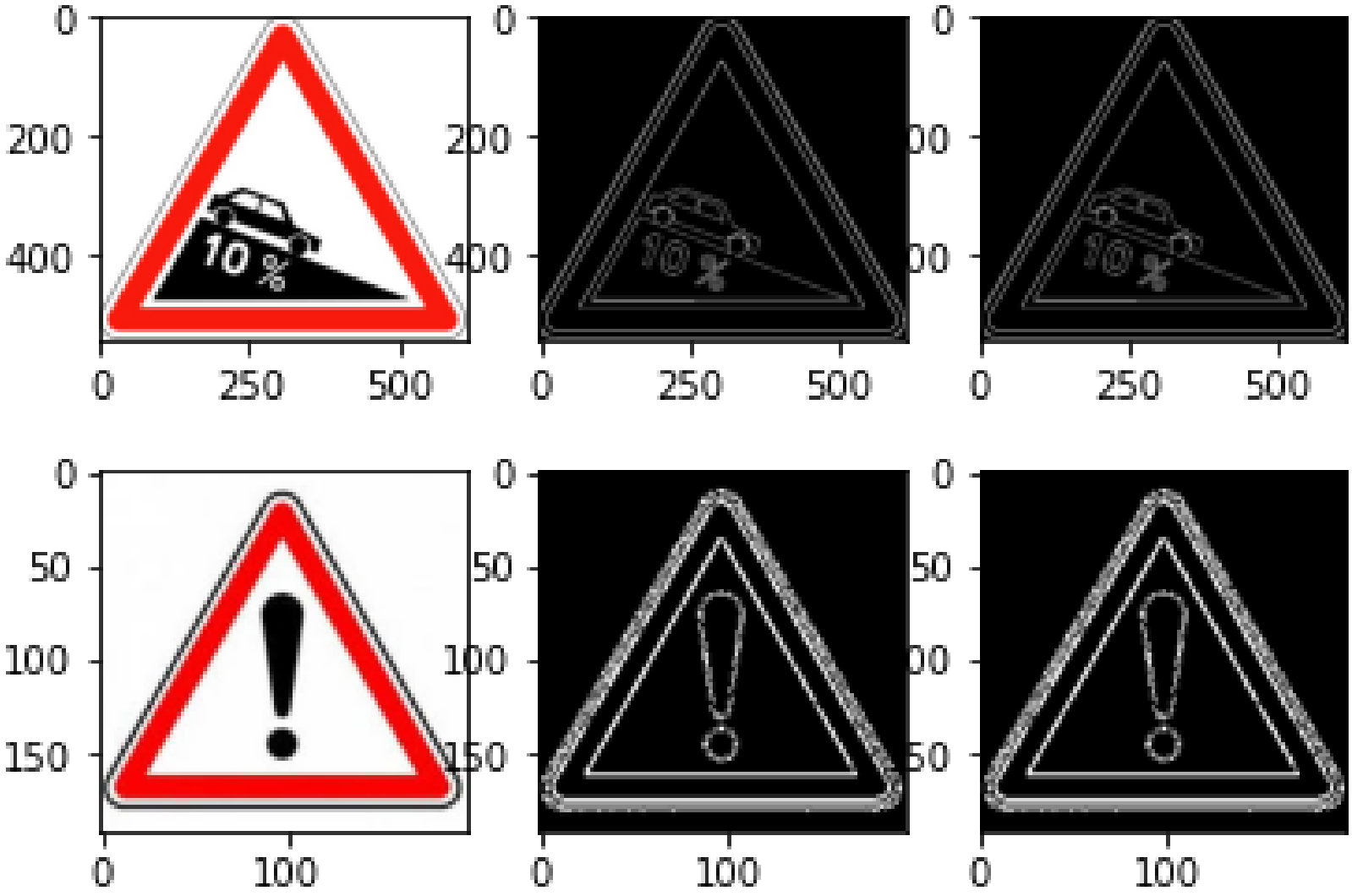
Niveaux de gris, dynamique et contours:

```python
def niveaux_gris(M):
    n, m, k = M.shape
    A = np.zeros((n,m), dtype = np.uint8)
    for i in range(n):
        for j in range(m):
            R,V,B = M[i,j]
            A[i,j] = R//3 + V//3 + B//3 #moyenne pour saisir les points A[i,j] de la matrice
    return A

def dyn(M):
    """
    Augmente au maximum la dynamique d'une image en niveaux de gris.
    """
    M2 = np.array(M, dtype=np.float64)
    mini = np.min(M)
    maxi = np.max(M)
    return np.array((M2 - mini)*255/(maxi - mini), dtype=np.uint8)


def contours(M):
    n, m = M.shape
    Mc = np.array(M, dtype=np.int32)
    M1 = np.zeros((n,m), dtype=np.int32)
    M2 = np.zeros((n,m), dtype=np.int32)
    M1[1:-1, :] = Mc[2:,:] - Mc[:-2,:] #derivation = filtrage passe-haut
    M2[:, 1:-1] = Mc[:,2:] - Mc[:,:-2]
    M3 = np.abs(M1) + np.abs(M2)    #norme du grad de l'intensite lumineuse
    M4 = np.array(M3, dtype=np.uint8)
    return M4
```

Exemple de dynamique et de **contours** pour un panneau danger et attention:



Pour pouvoir appliquer la transformée de Hough à une image, il faut préalablement avoir sélectionné des points caractéristiques de l'image (des points de variation de contraste par exemple).

Pour repérer ces points, il existe par exemple des opérateurs informatiques qui calculent des moyennes pondérées sur les contrastes du pixel considéré et de ses voisins.

Il en existe une multitude qui ont des conséquences différentes, à utiliser selon les effets cherchés.

Exemple : l'opérateur Laplacien. Filtres de **Sobel** + gaussien.

Pour Sobel:

On se donne la matrice de M(x,y) de l'image et les matrice des gradients suivant x et y, respectivement :

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \; G_y = \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Les 2 matrices correspondant à l'opération de dérivation respectivement en x et en y sont alors

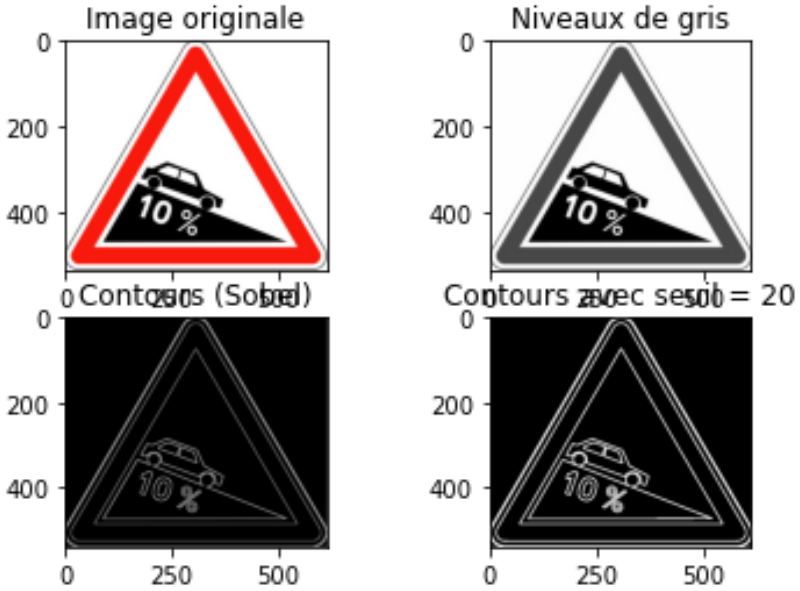
$$G1_x = G_x * M$$

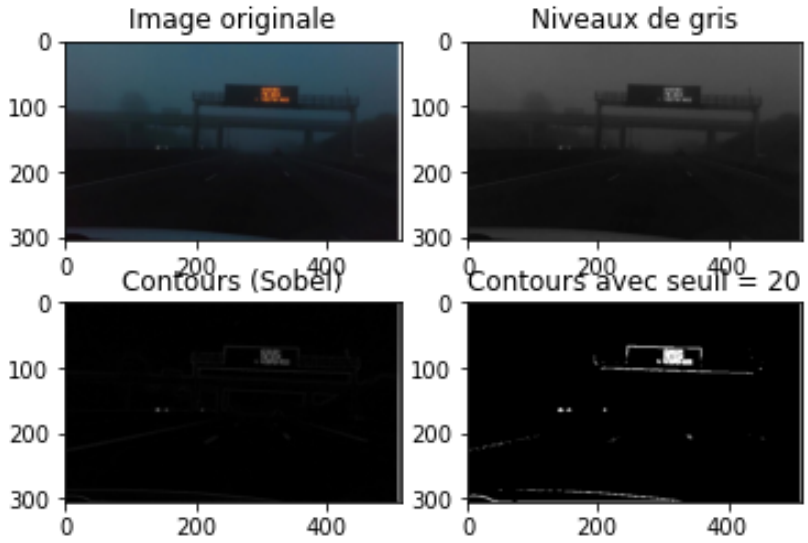
$$G2_x = G_y * M$$

On renvoie alors comme pour les contours le gradient de l'image:

$$G = \sqrt{G1_x^2 + G2_y^2}$$

Exemples avec le panneau danger et un panneau autoroutier :





Un paramétrage naïf

$y = ax + b$

Cependant ce choix de paramétrage est maladroit car les paramètres ne varient pas de façon uniforme. Par exemple, on ne peut pas représenter une droite verticale dans cet espace (car le coefficient directeur est infini).

Un programme informatique devrait alors faire des approximations pour représenter ces paramètres sans utiliser trop de mémoire.

Il est alors plus judicieux d'utiliser les coordonnées polaires :

Passage en coordonnées polaires

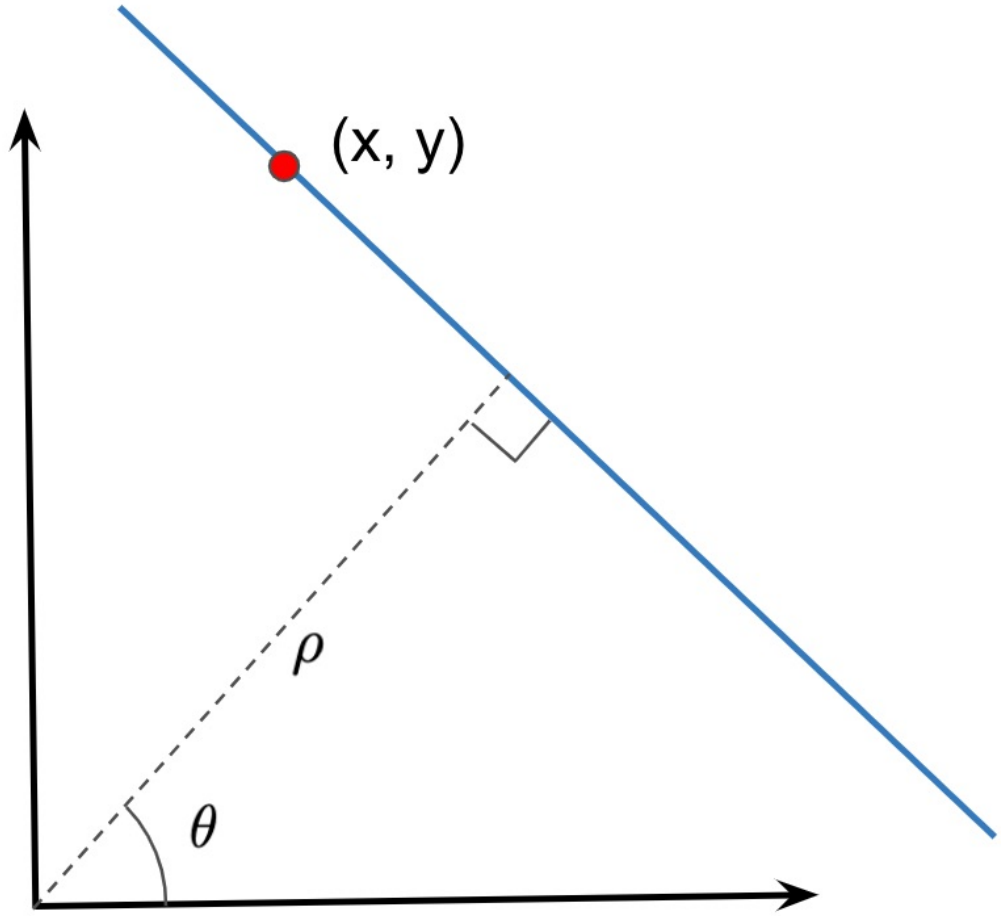
$$\rho = x cos\theta + y sin\theta$$

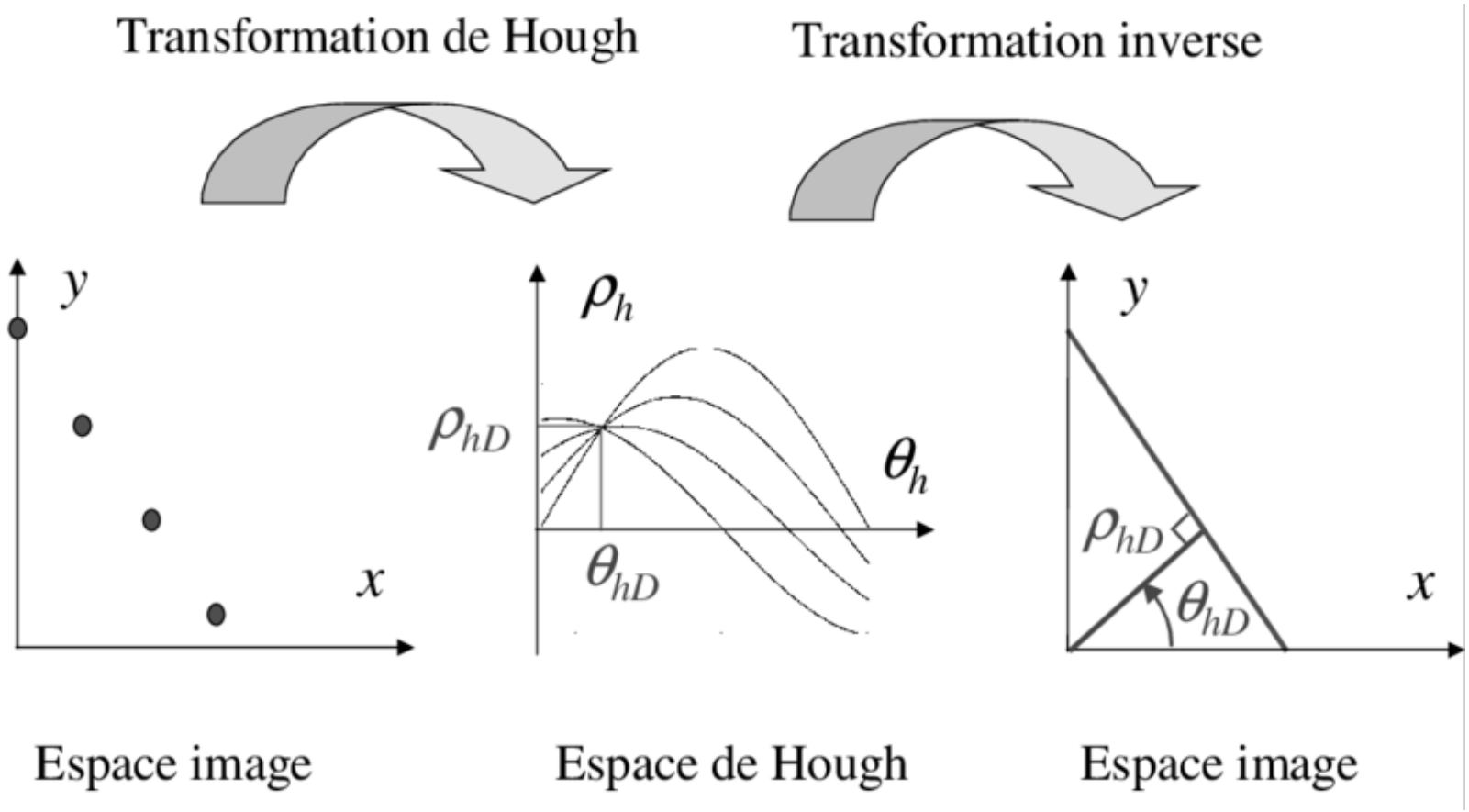
Ainsi, en balayant le paramètre sur l'intervalle $]-\frac{\pi}{2}, \frac{\pi}{2}]$ on obtient une sinusoïde dans l'espace de Hough, le paramètre θ variant, pour xmax et ymax fixés, sur l'intervalle [- $\rho_{max}; \rho_{max}$]*où* :

Balayage dans l'espace de Hough

$$\rho_{max} = \sqrt{x_{max}^2 + y_{max}^2}$$

Les variations de ρ en fonction de θ sont plus uniformes, et le paramètre est borné. Ainsi la représentation informatique de l'espace de Hough est facilitée.


(x, y)
ρ
θ



À chaque point à gauche est associée une sinusoïdale à droite. À chaque point à droite est associée une droite à gauche.
On voit apparaître à droite deux nuages de points noircis, qui correspondent aux 2 ensembles de droites que l'on distingue à gauche.

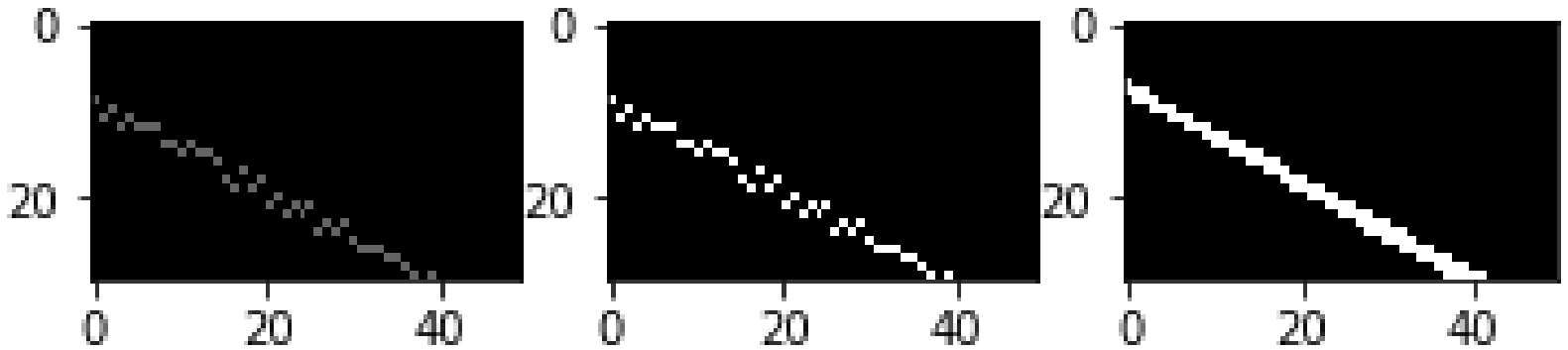
Avantages :

- Elle s'avère très efficace pour la détection de lignes et est peu sensible au bruit sur l'image

Inconvénients :

- Cette transformée de Hough ne retrouve que des droites et peut détecter de fausses droites ou des segments insignifiants

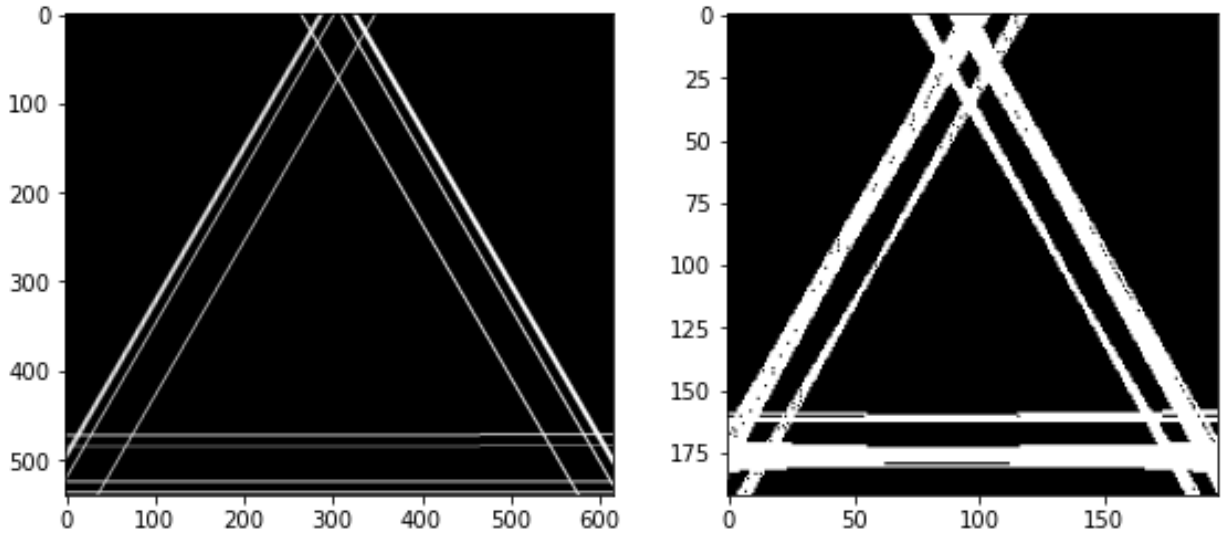
Exemple de transformée de Hough appliquée à une droite bruitée (programme hough.py en annexe):



Pour cela, on discrétise l'espace : on découpe l'espace en rectangles qui contiennent alors des maxima locaux de votes. Les amas de points d'une droite comptent alors tous pour le même accumulateur (à condition d'avoir choisi une discrétisation assez large).

Ainsi, on perd en précision sur la nature de la droite (la précision pour chaque paramètre dépendant de la taille des cases choisies), mais on y gagne en détection et en mémoire (le tableau prend moins de place, et chaque case reçoit des votes de façon plus significative).

Exemple appliqué au panneau danger et attention (programme hough.py en annexe)



La transformée de Hough ne se limite pas seulement aux droites. En effet, elle utilise seulement les paramètres représentatifs du motif que l'on cherche. Ainsi, on peut généraliser la transformée à toute courbe dont on connaît une équation paramétrique. Il suffit de faire une bijection de ces surfaces à n paramètres sur un espace de dimension n.

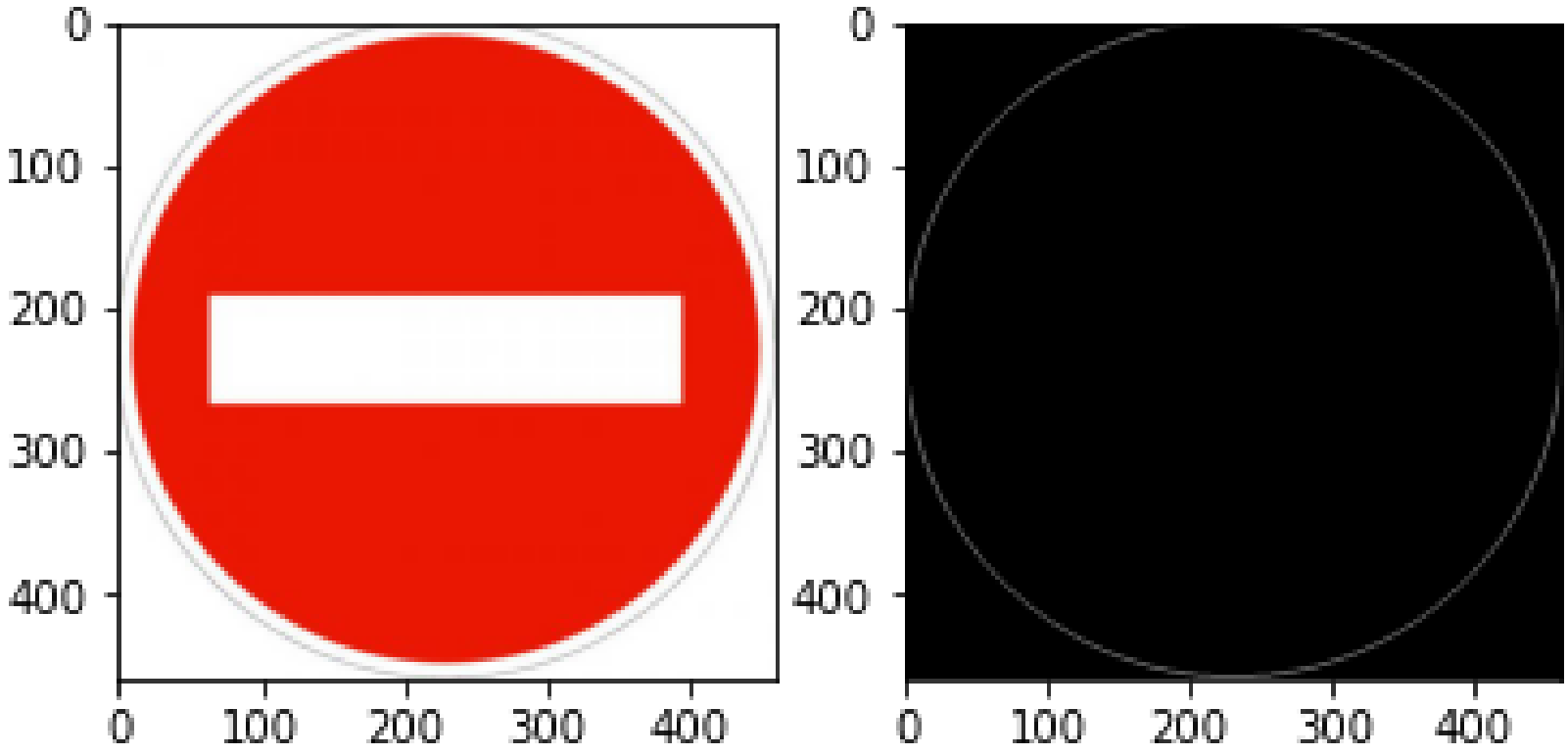
Par exemple, la transformée de Hough est performante dans la recherche de cercles d'un rayon donné :

Equation d'un cercle de centre (a;b) et de rayon r

$(x-a)^2 + (y - b)^2 = r^2$

Chaque point de (x;y) de l'image devient un cône dans l'espace des paramètres (a,b,r). Pour un rayon fixé, on obtient un cercle.

Exemple de la transformée de Hough appliquée au panneau sens-interdit (programme hough circle detection.py en annexe):



Algorithme en pseudo code

Accumulation grâce aux votes pour les droites

Début

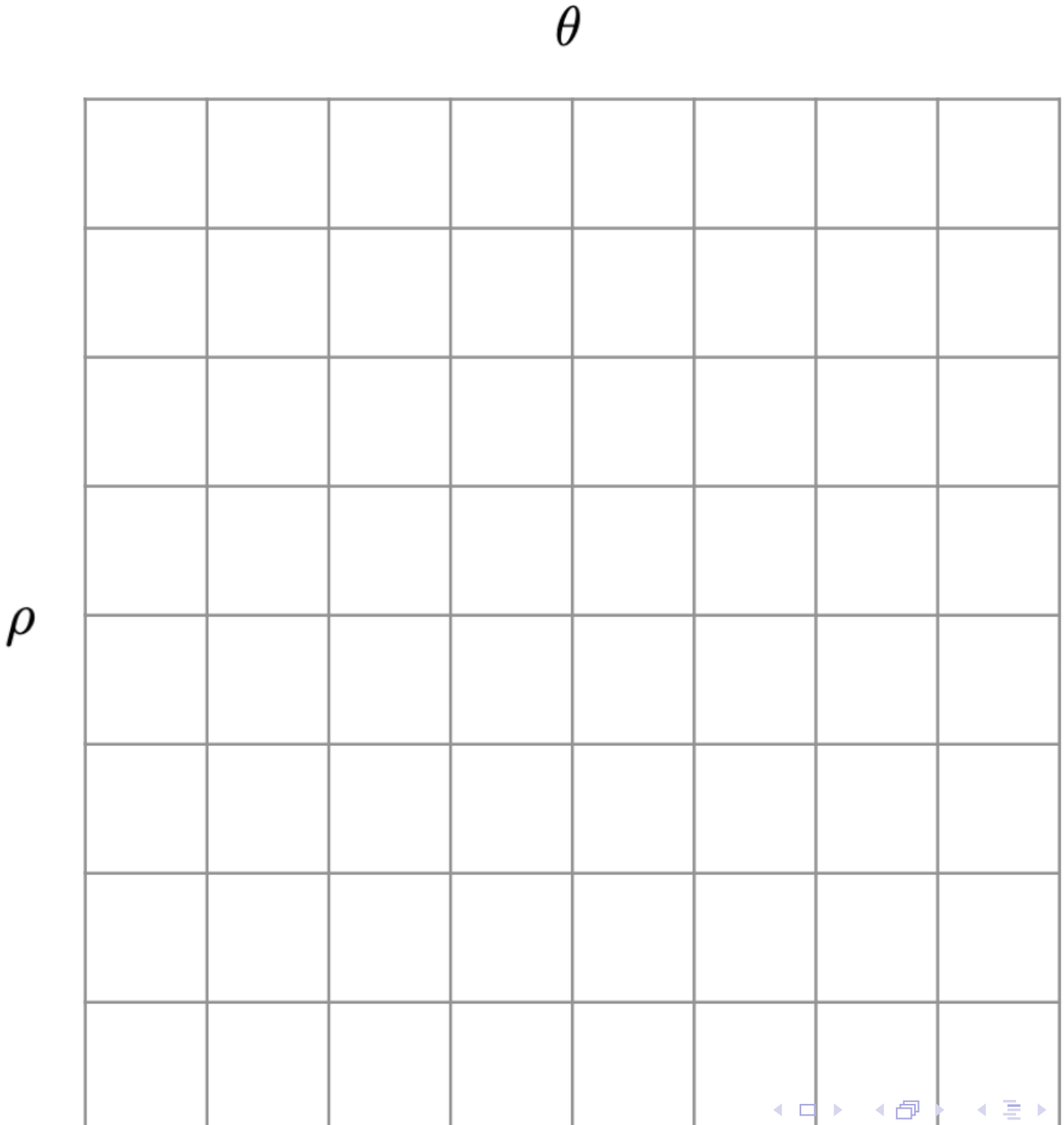
-Quantifier l'espace des paramètres avec un maximum et un minimum pour les 2 paramètres.

-Initialiser un tableau à 2 entrées à 0.

Pour chaque point (x;y) détecté par l'opérateur de détection de contours, pour θ variant entre ses extrema :

-Incrémenter la case (θ , $\rho =$ arrondi (x.cos(θ)+y.sin(ρ)))

Fin


θ
ρ

Algorithme en pseudo code

Seuillage pour completer la détection

Pour tous les couples $(\theta;\rho)$ considérés comme droites après accumulation

Si Macc[ρ, θ] $\geq$ *seuil* :

Ajouter $(\theta;\rho)$

III

Pratique en conditions réelles

Sur la route, l'ensoleillement, la pluie, et les différentes conditions météorologiques empêchent une détection optimale. On peut y remédier en :
-sélectionnant un meilleur filtre
- augmentant le contraste :

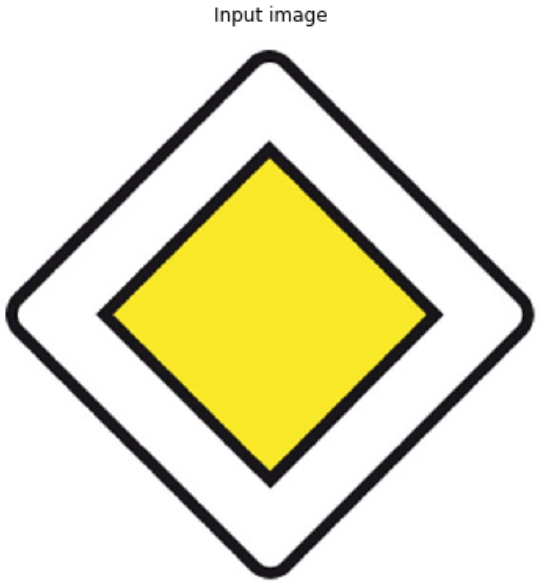
Contraste par étalement de l'histogramme (5 %):

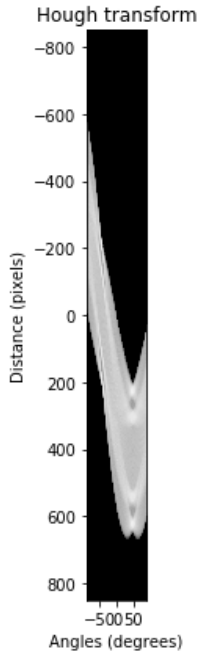
IV

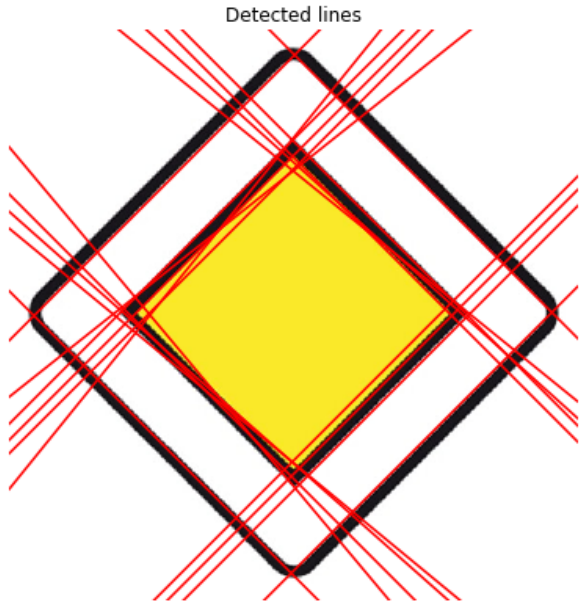
Comparaison des résultats entre transformée de python (non seuillée) et l'algorithme (seuillage)

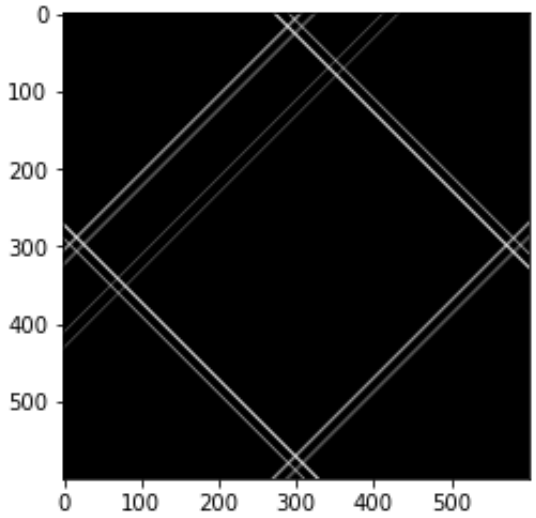
Comparaison de résultats entre la transformée de python (seuillage fixé) à notre transformée au seuil modifiable :
Si la transformée du module SKI MAGE de python est nettement plus précise pour les droites aux forts contours ; elle l'est beaucoup moins dès l'apparition d'irrégularités d'où un compromis nécessaire
.

Comparaison entre transformée sans seuillage et algorithme avec seuillage

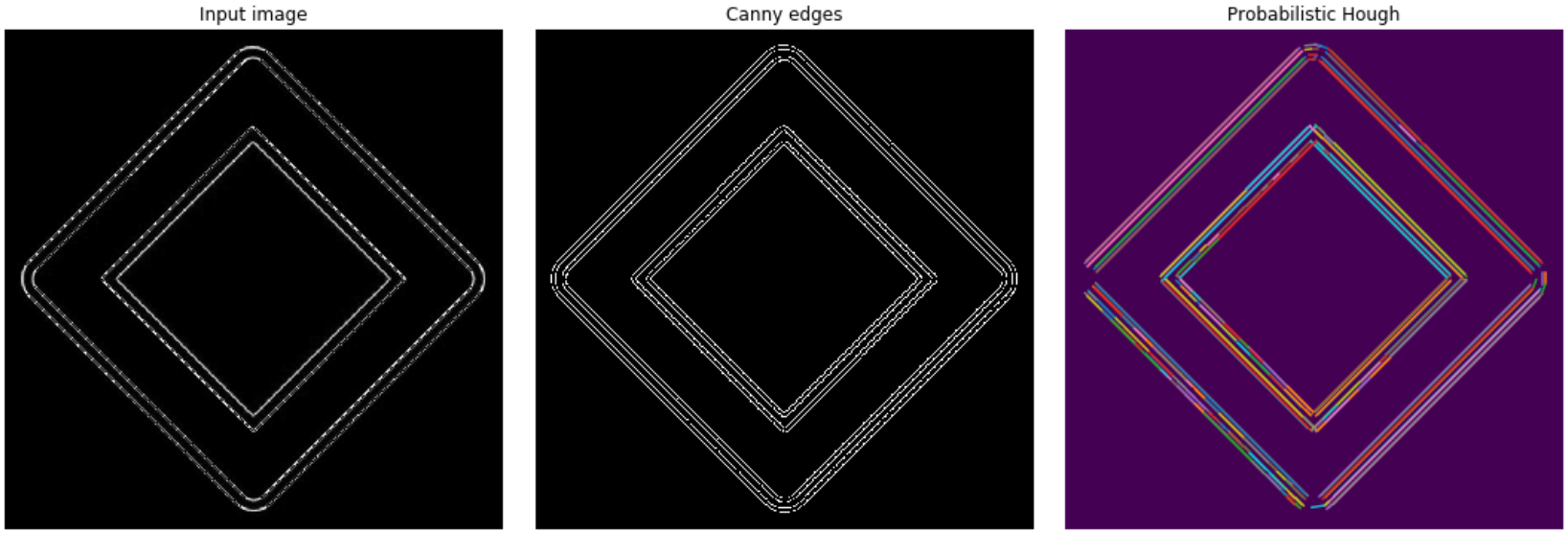








Toutefois , SKIMAGE dispose de la transformée probabiliste très précise . Au lieu d'appliquer la transformée de Hough à tous les pixels d'un contour, la THP se contente d'en sélectionner un pourcentage aléatoirement et de n'appliquer la transformée qu'à cet échantillon :



IV

Optimisation des résultats en conditions réelles

Tests à faire : jouer avec le contraste et les filtres sur des panneaux pris de nuit ou sous un mauvais éclairement (humidité...)

Envoyé