EL RAIS David CHANSUK Chatri

Rapport de Projet

Analyse d’une image contenant un escalier

Dans le cadre du master informatique 1ère année de l'Université de Paris, il nous a été donné pour objectif de réaliser un système capable de détecter et calculer le nombre de marche d'un escalier.

Concernant les critères de conception, le langage de programmation utilisé pour réaliser l'ensemble du projet est C++. L'API utilisée pour implémenter l'interface graphique est **Qt.**

Le projet contient deux dossiers permettant de lancer l'application, le premier **code/** permet de tester l'application en utilisant les méthodes 1 à 3 via l'interface graphique et le second **codeExecutable/** permet d'utiliser toute les méthodes en exécutant manuellement les fichiers sources sans avoir besoin d'installer Qt.

L'ensemble des images utilisé dans ce rapport est disponible dans le dossier **data/images.**

Nous allons utiliser l'image **b.jpg** pour appuyer nos propos et faciliter la compréhension du rapport.

La distinction se fait à partir des contre-marches donc un pas est constituer d'une contre-marche et d'une marche.

Ce rapport retrace les travaux et les choix effectués durant ce projet.

Sommaire

I. Première Méthode

1. Principe de la méthode

La première méthode consiste en un balayage vertical et horizontal sur l'image, avant d'entrer dans le vif du sujet, nous devons tout d'abord faire un pré-traitement sur l'image afin d'obtenir les meilleurs résultats.

2. Pré-Traitement

Image 1. Image témoin composée de 8 marches.

Travailler avec trois canaux de couleurs (bleu, vert, rouge) ne nous apporte pas d'information alors on converti l'image en nuance de gris.

Image 2. Conversion de l'image en nuance de gris.

On applique ensuite un **threshold binaire** sur l’image, ceci nous permet de bien faire la distinction entre les marches et les contre-marches, la valeur du threshold est déterminer de manière dynamique en fonction de l’image (50% de pixel noir - 50% de pixel blanc).

A ce stade nous avons des rectangles noirs avec plus ou moins de bruit en fonction de l’image.

On applique donc un filtre médian pour réduire le bruit.

Image 3. Application d’un threshold binaire et d’un filtre médian.

3. Déterminer le nombre de marche

A. Parcours Vertical

Pour déterminer le nombre de marche, on utilise un balayage vertical, c’est à dire que l’on parcourt l’image verticalement.

Pour chaque parcours, on incrémente une variable correspondant au nombre de pixel noir rencontré consécutivement, si le pixel courant est blanc et que le nombre de pixel noir rencontré est supérieur à un seuil, alors on affiche un cercle et on incrémente le nombre de marche vue durant le parcours sinon on remet à zéro le nombre de pixel noir rencontré.

A la fin du parcours, on stocke le nombre de marche dans un tableau, on fait de même pour chaque parcours.

Pour déterminer la valeur à retourner, on trie le tableau dans l'ordre décroissant, ensuite on compare deux à deux les valeurs maximal et on retourne l'élément maximal qui à la plus grande fréquence, ce qui correspond au nombre de marche dans l'image.

Image 4. Résultat du parcours vertical, on obtient 8 marches.

B. Parcours Horizontal

On parcours l'image horizontalement, on s'intéresse donc aux lignes qui définissent une marche.

Une marche est une succession de ligne noir, suivi d'une succession de ligne blanche, nous devons donc rechercher ce pattern dans nos images.

Pour chaque parcours, on initialise à zéro une variable correspondant au nombre de pixel noir dans une ligne. Si le nombre de pixel est supérieur à un certain seuil alors la ligne est considère comme étant une ligne appartenant potentiellement à une marche.

On continue ce principe tant que l'on n'est pas sur une ligne toute blanche ou que l'on n'a pas parcourus l'ensemble de l'image.

Si le nombre de ligne appartenant potentiellement à une marche est supérieur à un certain seuil alors on incrémente le nombre de marche détecter dans l'image et on remet à zéro le nombre de ligne appartenant potentiellement à une marche et on continue le parcours sur le reste de l'image.

Image 5. Résultat du parcours horizontal, on obtient 8 marches.

4. Avantage et Inconvénients

Concernant les points forts de cette méthode, lorsqu'il s'agit d'un escalier droit avec ou non la présence de bruit (luminosité, présence de motif) l'algorithme arrive bien à détecter le nombre de marche.

Cependant le pré-traitement n'est pas suffisant pour détecter la position des marches dans l'image.

De plus, plusieurs dépendances sont à noter au niveau des méthodes de parcours de l'image.

En effet, le balayage vertical est soumis à un seuil afin de déterminer si une marche est bien une marche ou un bruit.

On doit aussi souligner le pas entre deux balayages, en d'autre terme la distance qui sépare deux parcours verticaux, si il est trop petit on risque dans certains cas de prendre en compte du bruit donc de faussé nos résultats et si il est trop grand on risque de passer à côté du bon nombre de marche, ce qui peut avoir une grande influence sur les escaliers tournants.

La méthode horizontale est elle aussi soumis à plusieurs dépendances, en effet, le nombre de pixel noir qu'une ligne doit avoir pour être considérée comme une ligne appartenant potentiellement à une marche ainsi que le nombre de ligne successive pour qu'un ensemble de ligne soit considérée comme une marche sont soumis à un seuil.

II. Deuxième Méthode

1. Principe de la méthode

La seconde méthode est une amélioration de la première en ce qui concerne la détection de la position des marches dans l'image, nous utiliserons comme précédemment le même balayage vertical et horizontal.

2. Pré-Traitement

Nous avons pu remarquer que la luminosité ne nous permet pas d'avoir une bonne représentation et nous devons utiliser d'autre méthode de pré-traitement afin de reconstituer les rectangles correspondant aux contre-marches.

L'idée est de réduire le bruit c'est-à-dire supprimer les pixels noirs qui n'apporte aucune information.

On supprime (met en blanc) les lignes ayant un ratio (compte le nombre de pixel noir divisé par la taille de la ligne) de pixel noir inférieur à un certains seuil. On fait de même avec les colonnes.

Ensuite on essaye d'avoir des lignes complètes c'est-à-dire ne pas avoir des clusters de pixels noirs sur une même ligne, pour cela on détermine le premier et le dernier pixel d'une ligne et on met en noir tout les pixels qui se situe entre eux.

Enfin, on va appliquer une méthode locale, c'est-à-dire qu'on modifie la couleur d'un pixel en fonction de la couleur de ses voisins.

Le principe est le suivant : Si le pixel courant est blanc et que ses voisins sont noirs, alors le pixel devient noir.

Pour cela, on parcours l'image ligne par ligne, si on est sur un pixel blanc alors on vérifie si le pixel du **haut** et celui de **gauche** ou du **haut** et celui de **droite** ou du **bas** et celui de **gauche** ou du **bas** et celui de **droite** est noir si c'est le cas alors on change la valeur du pixel en noir.

On applique cette méthode locale qu'une seul fois car si on applique cette méthode plusieurs fois sur un escalier tournant les contre-marches étant de forme triangulaire, on perdrait toute l'information et on se serait retrouvé avec un grand bloc noir représentant l'escalier.

En utilisant toute ces méthodes de pré-traitement on a réduit le bruit.

On peut maintenant déterminer le nombre de marche.

Image 6. Image après pré-traitement

3. Déterminer le nombre de marche

Comme dit précédemment, on utilise les mêmes méthode pour calculer le nombre de marche.

Image 7. Résultat du parcours vertical, on obtient 8 marches.

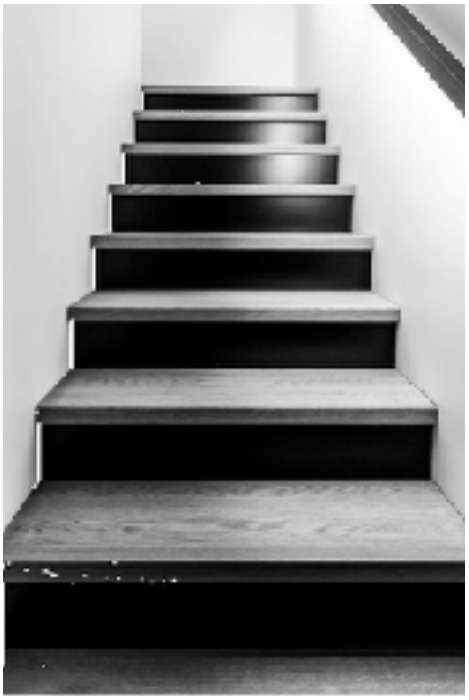

Image 8. Résultat du parcours horizontal, on obtient 8 marches.

4. Avantage et Inconvénients

Concernant les points forts de cette méthode, lorsqu'il s'agit d'un escalier droit avec ou non la présence de bruit (luminosité, présence de motif) l'algorithme arrive bien à détecter leurs positions.

Cependant dès que l'escalier tourne un peu ou que la prise de vue n'est pas face à l'escalier, donc la contre marche devient triangulaire et non plus rectangulaire on obtient des résultats moins précis.

Concernant les dépendances, lors du pré-traitement on supprime des lignes/colonnes ayant un ratio de pixel noir faible, le ratio est déterminer par tâtonnement entre les différentes images de notre base de donnée.

III. Troisième Méthode

1. Principe de la méthode

La troisième méthode consiste en un balayage vertical et horizontal, avec comme idée générale de détecter les positions des contre-marches et de les redessiner sur une nouvelle matrice afin de supprimer tous les bruits via les fonction prédéfinies dans OpenCV.

2. Pré-Traitement

Nous appliquons le même principe que précédemment, c'est-à—dire que nous transformons en niveau de gris, nous appliquons un threshold binaire.

Dans le but de supprimer le bruit on utilise un filtre médian et un algorithme de débruitage des moyennes non locaux.

Image 9. Suppression du bruit avec filtre médian et algorithme de débruitage.

Ensuite dans le but de redessiner les contre-marches nous devons détecter les contours des contre-marches pour cela on utilise un filtre de Canny.

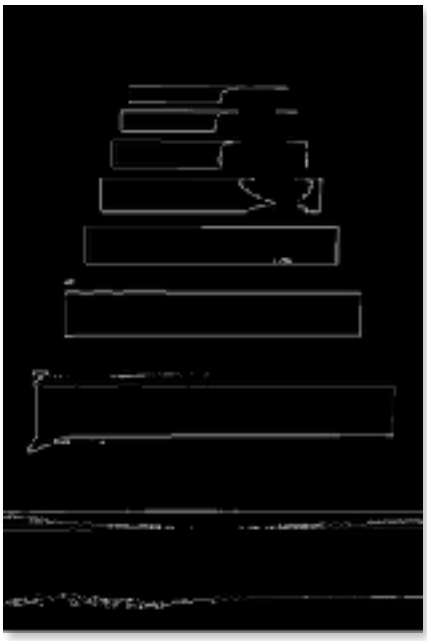

Image 10. Application du filtre Canny.

Une fois que nous avons les contours nous devons récupérer les coordonnées des premiers et des derniers pixels sur chaque ligne et les afficher sur une matrice noir.

Image 11. Détection du premier et du dernier pixel

On va ensuite tracer une ligne entre les premiers et les derniers pixels.

Image 12. Traçage de ligne entre les premier et derniers pixels de chaque ligne.

Enfin, on trace les rectangles à partir des résultats obtenus.

Image 13. Détection des rectangle à partir des résultats précédent

3. Déterminer le nombre de marche

Pour déterminer le nombre de marche, un balayage vertical et horizontal est effectué.

Image 14. Résultat du parcours vertical, on obtient 8 marches.

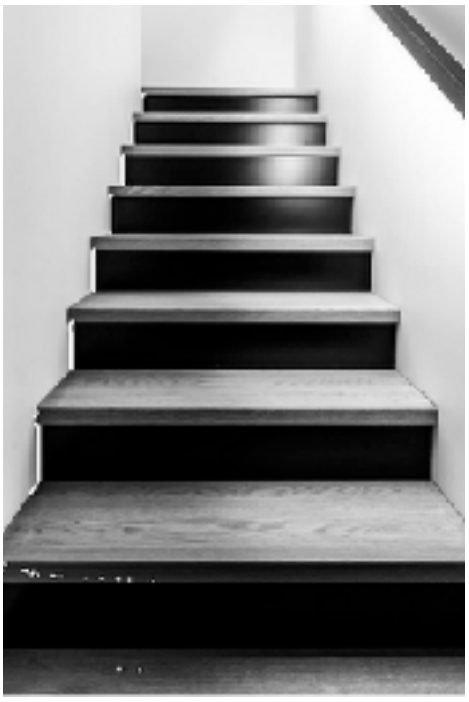

Image 15. Résultat du parcours horizontal, on obtient 10 marches.

4. Avantage et Inconvénients

Concernant les points forts de cette méthode, elle permet d'avoir un autre point de vue concernant le pré-traitement avec plusieurs méthodes connues pour leur efficacité.

Les points faibles sont plus nombreux, ce type de méthode ne s'applique que dans le cas d'escalier qui ne tourne pas car les contre-marches doivent être rectangulaire.

De plus, cette méthode est sensible au bruit notamment la luminosité.

Il n'y a quasiment pas de dépendance dans cette méthode, en effet on peut seulement citer la distance à respecter entre deux pixels consécutif (lors du traçage de ligne entre le premier et le dernier pixel) pour que le second pixel ne soit pas supprimé car considéré comme du bruit.

IV. Quatrième Méthode

1. Principe de la méthode

Pour cette méthode la détection des marches se basera avant tout sur les transformations de **Hough** (Probabiliste et standard) afin de nous permettre d'obtenir les lignes formées par les angles des marches.

2. Pré-Traitement

A. Transformation Morphologique

Dès le chargement de l'image il faut avant tout manipuler l'image pour camoufler tous bruits ou éléments pouvant être compromettant à la détection de ligne.

Les principales difficultés seront les contremarches et le nez d'escalier pouvant rajouter des lignes supplémentaires à l'identification des marches.

Pour ce faire, nous travaillerons sur des transformations morphologiques nous permettant d'être plus flexible sur les images en entrée. Voici les principales transformations tirées de la documentation :

- **Erosion :** Tous les pixels aux limites du noyau (kernel) seront rejetés selon la taille de ce dernier, ainsi l'épaisseur de l'objet au premier plan rétrécira ou encore la zone blanche de l'image se voit diminué ce qui est utile pour éliminer les bruits de couleurs blancs.

- **Dilatation (dilation) :** À l'opposé de l'érosion, la zone blanche ou l'objet en premier plan verra sa taille augmenté.

- **Ouverture** (**opening**) **:** un autre nom pour désigner une érosion suivie d'une dilatation, ce qui est utile pour éliminer les bruits.

- **Fermeture (closing) :** La fermeture est l'inverse de l'ouverture, une dilatation suivie de l'érosion. Il est utile pour fermer de petits trous à l'intérieur d'un objet au premier plan ou de petits points noirs sur l'objet.

On peut aussi sélectionner la taille du noyau (jusqu'à 21) et sa forme (rectangulaire, forme de croix ou ellipsoïdale). Et le tout, combiné sous une forme spécifique : rectangulaire, forme de croix ou ellipse.

Image 16. Image témoin composée de 13 marches

Image 17. Image Morphologique(Closing – Rectangle – Kernel size 10)

Image 18. Image Morphologique (Closing – Ellipse – Kernel size 13)

B. Lignes Horizontal

De ce fait, les transformations permettent entre autres d'obtenir une matrice spécifique à la structure sélectionnée, on peut ainsi obtenir les lignes horizontales isolées.

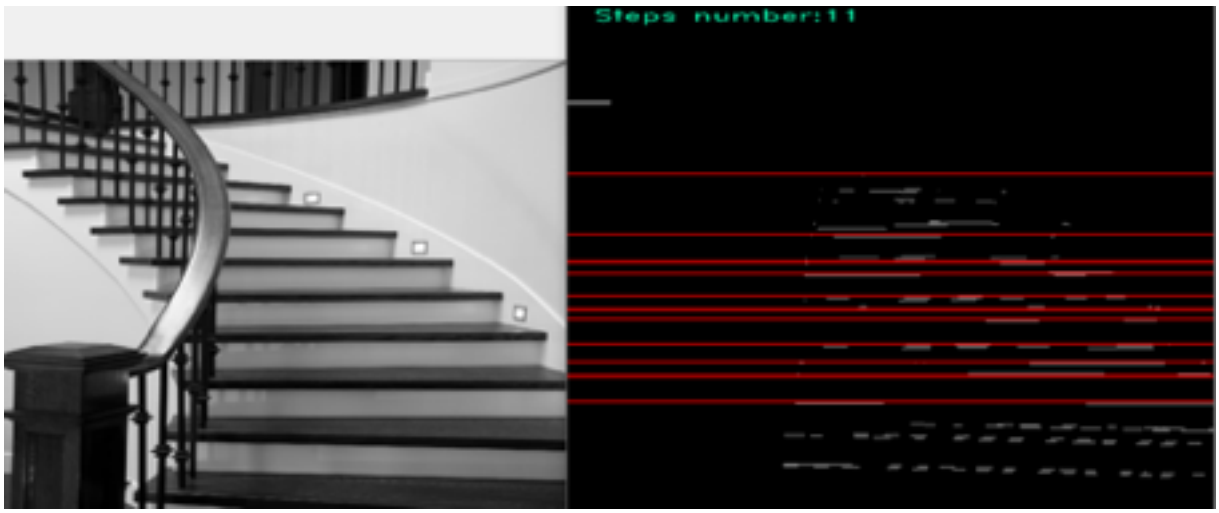

Image 19. Détection des lignes horizontale.

C. Canny

Avant de poursuivre avec la détection de ligne par Hough, on utilise un filtre de Canny car il est nécessaire de déterminer le contours de notre objet au sein de l'image.

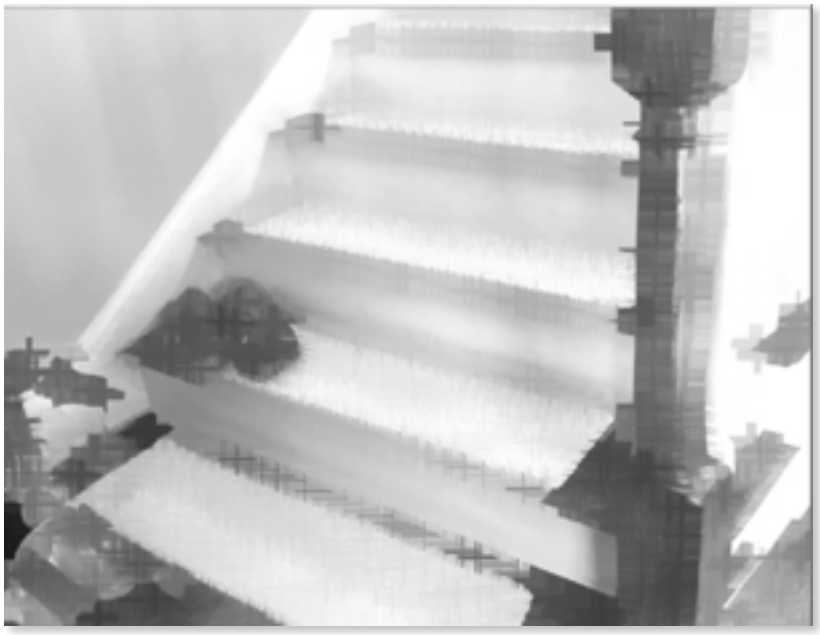

Image 19. Utilisation de la fonction Canny

3. Determiner le nombre de marche

Nous obtenons alors des images transformées sur lesquels la transformation de Hough peut déterminer des lignes existantes.

Pour les résultats suivants, un étalonnage de length gap et threshold nous approche à peu près d'une reconnaissance proche à la vérité terrain.

Concernant les résultats, cette méthode étant facilement talonnable, il faudra éviter d'obtenir un ajustement pour seulement obtenir aveuglement le nombre de marche réel.

Les paramètres strictement nécessaires à une bonne évaluation sont :

- Le calcul du nombre de contremarche ou de nez de marche selon les photos.

- L'exclusion des lignes encadrant l'image, faussant le résultat (cf image)

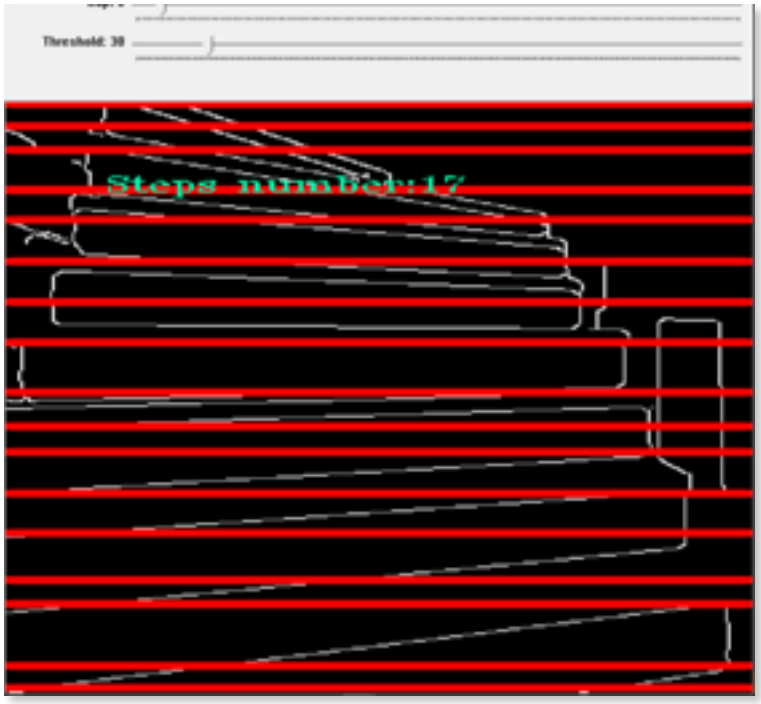

Image 20. Exemple de ligne supérieur et inférieur encadrant l'image

Lorsque le processing est satisfaisant (en gris après transformation morphologique ou filtre horizontal), la matrice en sortie est appliquée à la transformation de Canny.

La matrice obtenue indique dorénavant les contours des éléments restants observés dans l'image. On reconvertit la matrice de contours en 2 matrices de couleur :

- Une pour le contrôle des lignes de la transformé probabiliste
- Une pour la transformé standard.

On a dorénavant, tous les éléments nécessaire pour appliquer la transformation de Hough.

La transformation de Hough probabiliste nous renvoie un vecteur de point en 4 dimensions. On y affiche les lignes obtenus sur la matrice de contrôle.

On applique à nouveau un algorithme sur le même vecteur :

- Le premier point du vecteur est stocké dans une variable, elle est ajouté à un nouveau vecteur d'entier.

- Chaque point enregistré dans le vecteur à 4 dimensions est comparé au vecteur d'entier simple, si leur différence est supérieur à 10 (une valeur arbitraire).

Une ligne est tracée et on incrémente alors notre compteur de marche.

Voici quelques résultats en exemple.

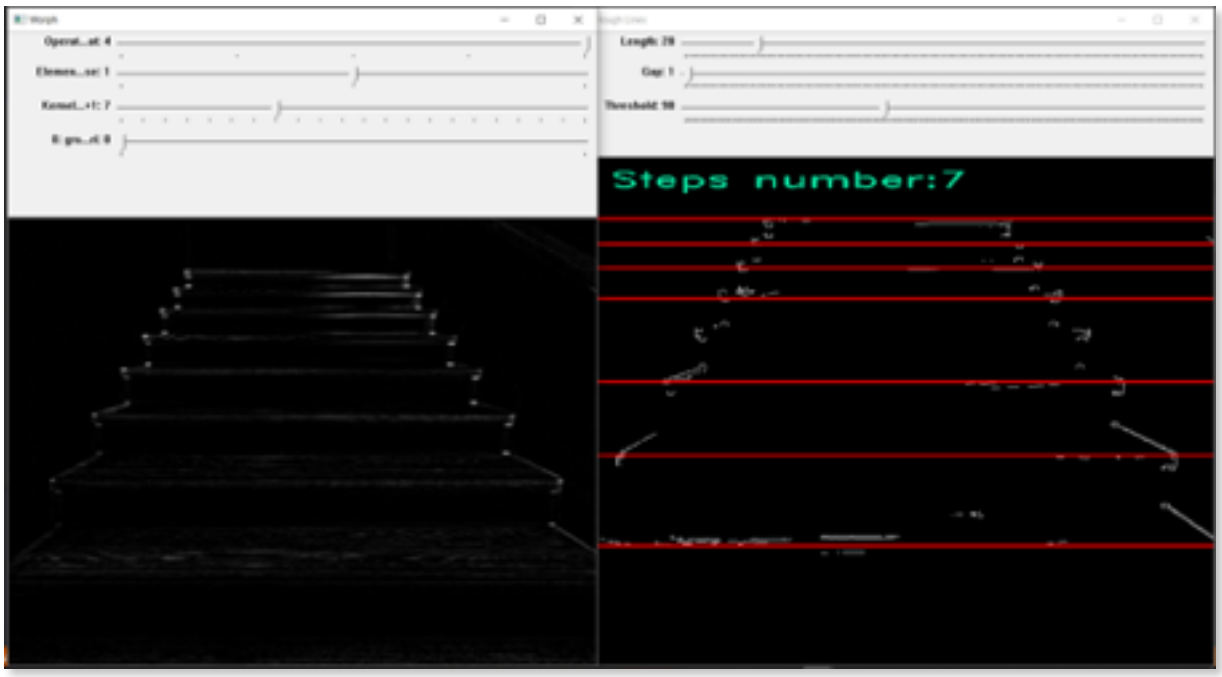

Image 21. Détection de 7 marches sur 8

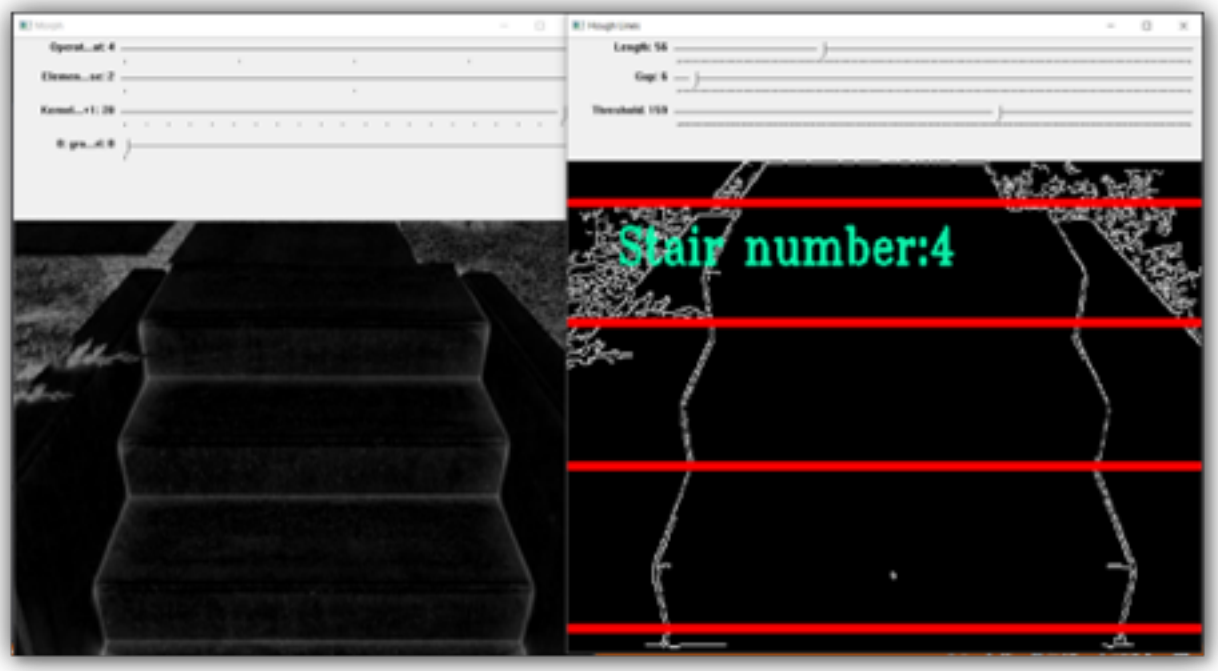

Image 22. Détection de 4 marches sur 3

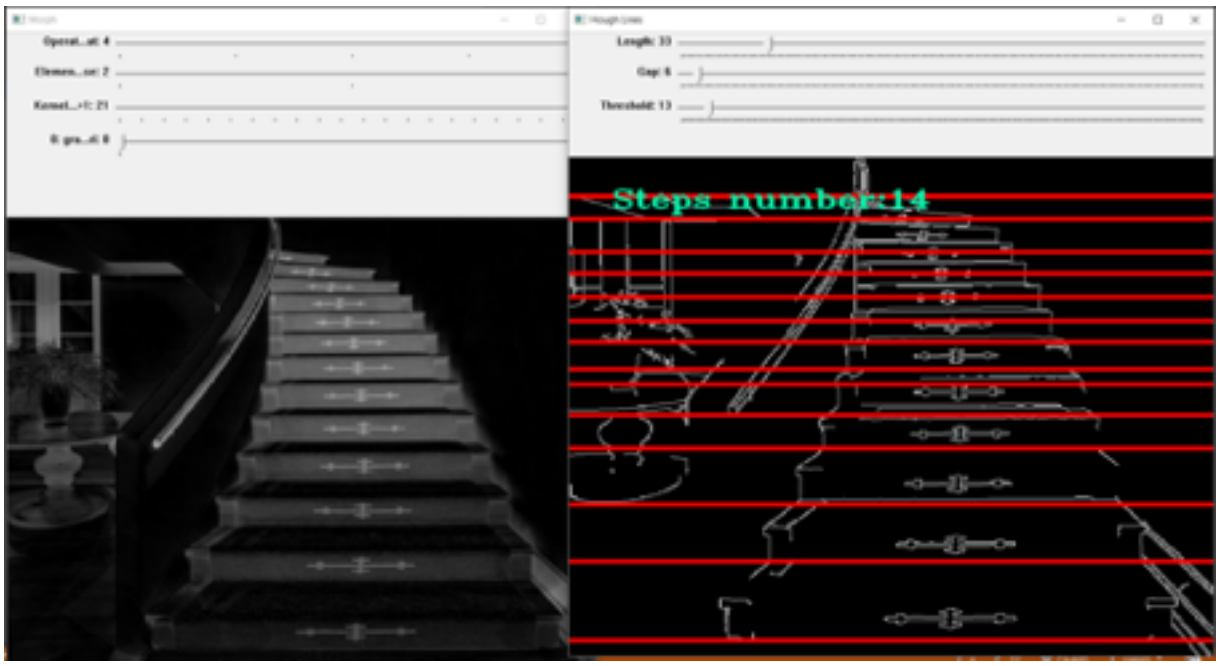

Image 23. Détection de 14 marches sur 13

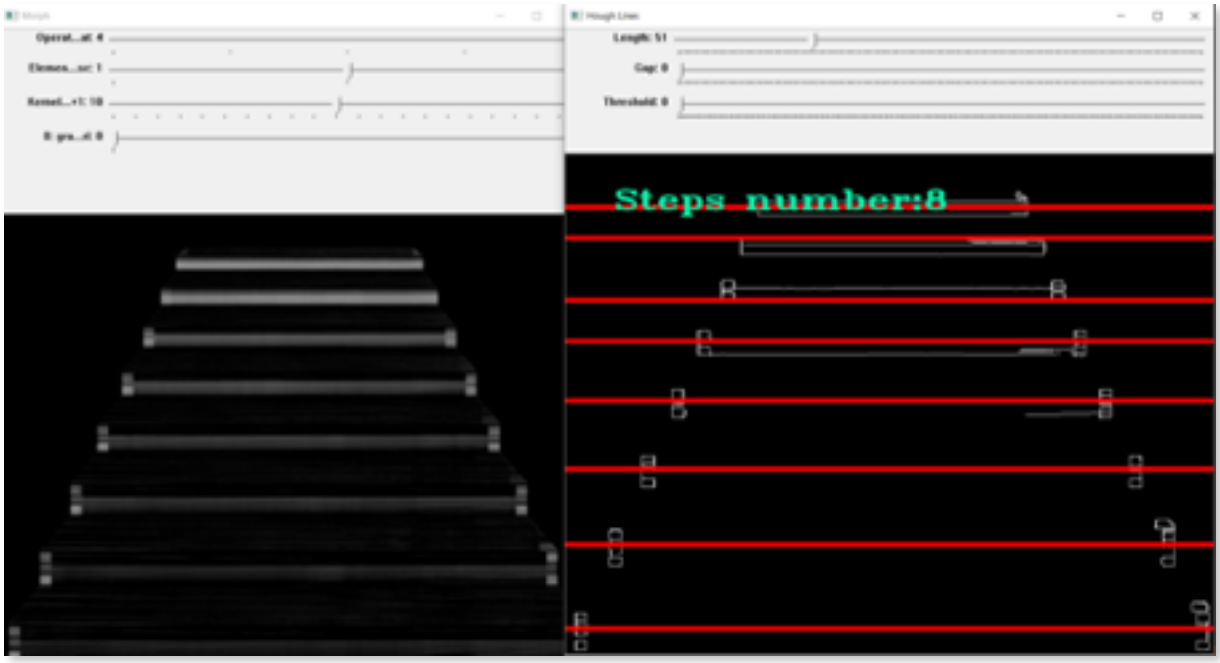

Image 24. Détection de 8 marches sur 8

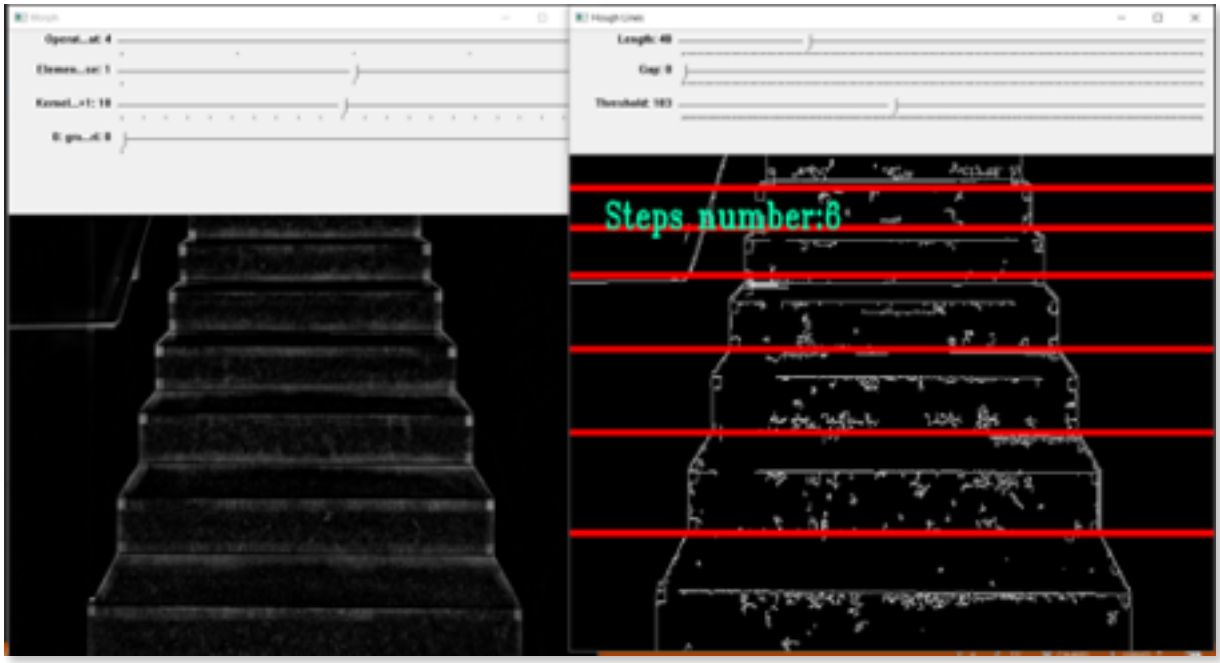

Image 25. Détection de 6 marches sur 6

4. Avantage et Inconvénients

Les résultats restent cependant des approximations, l'ajustement nous permets d'obtenir les valeurs qu'on cherche, il faut dans ce cas prêter attention à la caractéristique de la marche qu'on souhaite détecter.

On obtient de bons résultats concernant la suppression de bruit et la détection de forme.

Selon les paramètres la taille de l'image ou le nombre d'élément peuvent ne plus être suffisant pour y appliquer Canny ou Hough.

Les images dans une mauvaise orientation ou avec des bruits en grandes quantités fausse les résultats.

Les calculs peuvent cependant devenir vite couteux selon la sélection de forme : la forme ellipsoïdale notamment.

Pour des images avec des reflets : la morphologie black-hat est préférée.

Des images avec beaucoup de bruit : plutôt closing.

V.Conclusion et Perspective

Pour conclure l'ensemble du projet, nous devons avoir un regard global sur les différentes méthodes utilisé avec chacun leurs avantages et leurs inconvénients.

Nous allons étudier les performances de chaque méthode à travers leur capacité à détecter bon nombre de marche.

Prediction / Reel	Méthode 1	Methode 2	Methode 3	Methode 4
a.jpg	H:8/8 - 100% V:8/8 - 100%	H:8/8 - 100% V:7/8 - 87%	H:8/8 - 100% V:8/8 - 100%	8/8 - 100%
b.jpg	H:8/8 - 100% V:8/8 - 100%	H:8/8 - 100% V:8/8 - 100%	H:10/8 - 75% V:8/8 - 100%	7/8 - 87%
c.jpg	H:6/6 - 100% V:9/6 - 50%	H:6/6 - 100% V:6/6 - 100%	H:0/6 - 0% V:7/6 - 83%	6/6 - 100%
d.jpg	H:3/8 - 37% V:6/8 - 75%	H:5/8 - 62% V:6/8 - 75%	H:8/8 - 100% V:7/8 - 87%	10/8 - 75%
e.jpg	H:5/8 - 62% V:3/8 - 37%	H:4/8 - 50% V:4/8 - 50%	H:2/8 - 25% V:3/8 - 37%	12/8 - 50%
f.jpg	H:0/13 - 0% V:10/13- 76%	H:2/13 - 15% V:2/13 - 15%	H:10/13 -76% V:5/13 - 38%	14/13 - 92%
g.jpg	H:0/4 - 0% V:6/4 - 50%	H:0/4 - 0% V:2/4 - 50%	H:3/4 - 75% V:2/4 - 50%	12/4 - 30%
h.jpg	H:0/8 - 0% V:7/8 - 87%	H:0/8 - 0% V:4/8 - 50%	H:7/8 - 87% V:6/8 - 75%	9/8 - 88%
i.jpg	H:0/11 - 0% V:10/11- 90%	H:3/11 - 27% V:5/11 - 45%	H:11/11-100% V:6/11 - 54%	22/11 - 20%
j.jpg	H:1/12 - 8% V:27/12 -25%	H:4/12 - 33% V:13/12- 91%	H:10/12 -83% V:18/12 -50%	Non Chargeable
k.jpg	H:2/3 - 66% V:3/3 - 100%	H:3/3 - 100% V:3/3 - 100%	H:2/3 - 66% V:1/3 - 33%	4/3 - 66%
Moyenne	H:43% V:71.8%	H:53.4% V:69.4%	H:71.5% V:64.3%	70.9 %

Tableau 1. Performance des méthodes : Détection du nombre de marche avec H : balayage Horizontal et V : balayage Vertical.

On remarque assez facilement que les méthodes ont quelques difficultés lorsqu'il s'agit d'escalier tournant, que les contre-marches ne sont pas assez large ou avec une forte présence de bruit (g.jpg).

On a une amélioration de la performance pour le balayage horizontal, ceci est dû à une meilleure représentation des contre-marches. Dans la première méthode on ne fait quasiment aucun pré-traitement dans la seconde on améliore la qualité on faisant un pré-traitement plus précis et dans la troisième on dessine nous même les contre-marches ce qui nous permet de n'avoir aucun bruit.

Par contre on a une baisse de la performance pour le balayage vertical, lors du pré-traitement on perd de l'information ce qui influe sur la performance.

En ce qui concerne la dernière méthode, les résultats résument parfaitement ce que nous avions dit au niveau des avantages et des inconvénients, les images avec une mauvaise orientation ou avec des bruits en grandes quantités fausse les résultats.

Le but de notre projet n'a pas été d'avoir un système qui à les meilleures performances mais de développer plusieurs méthodes et de comparer leur avantages et inconvénients.

Durant ce projet, nous avons implémenté plusieurs idées qui aurait pu à première vue augmenter les performances des méthodes, cependant utiliser une idée sur un type d'escalier ne fonctionne pas forcément sur l'ensemble de notre base de donnée et au contraire peut même faire diminuer les performances des méthodes, c'est pour cela que nous les avons pas implémenté dans ce projet.

Voici quelques exemples d'idées n'ayant pas été retenues.

Convertir l'image de BGR en HSV et ne pas travailler avec la partie Saturation pour diminuer la luminosité, cependant on se retrouvait avec une image plus sombre ce qui dans certains cas empêcher la bonne distinction entre les marches et les contre-marches.

Pour le balayage vertical, on aurait pu améliorer la méthode en faisant varier la valeur du seuil en fonction de la hauteur de l'image, les marches en haut de l'image sont le plus souvent plus petit que les marches du bas de l'image donc le seuil sera de plus en plus grand lors du parcours de l'image.

Ce qui aura pour effet de diminuer la dépendance à ce seuil, cependant lorsqu'on utilise cette méthode sur des image avec du bruit, le nombre de marche détecté devient rapidement faux, on a donc préférée fixé un seuil qui fonctionne bien, par tâtonnement.

Parmi les implémentations envisageables, nous pouvons utiliser un réseau de neurones afin de bien cibler l'escalier dans l'image et ne pas se soucier du bruit qui l'entoure.

Nous aurions aussi pu travailler avec les histogrammes de l'image, analyser leurs formes et faire apparaitre un pattern qui nous aiderait à déterminer le nombre de marche.

Enfin ci-dessous une suite de capture d'écran montrant l'utilisation de l'interface graphique Qt.

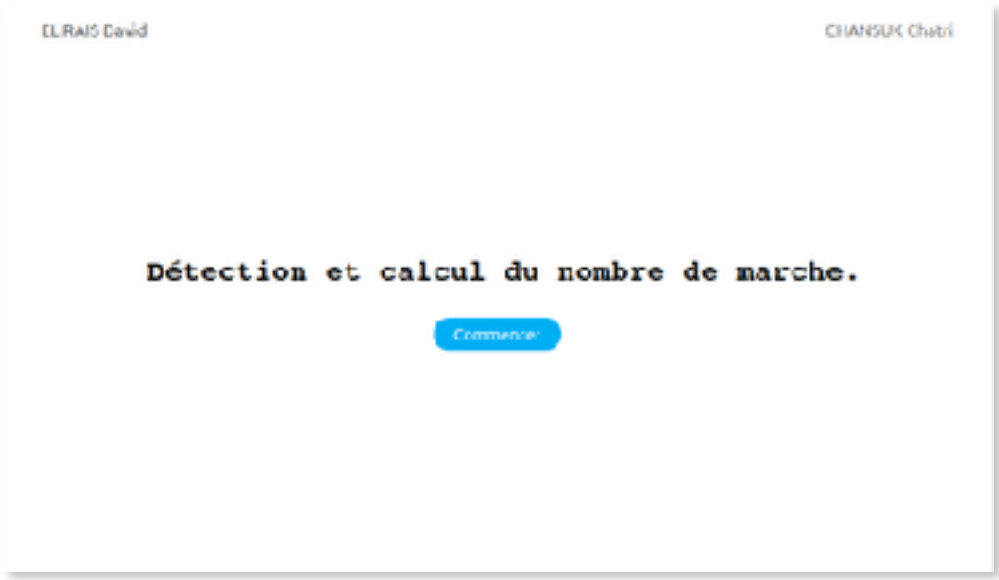

Image 26. Page d'accueil

Image 27. Chargement des fichier .jpg et .xml (1/2)

Image 28. Chargement des fichier .jpg et .xml (2/2)

Image 29. Méthode 1. Balayage Vertical

Image 30. Méthode 1. Balayage Horizontal

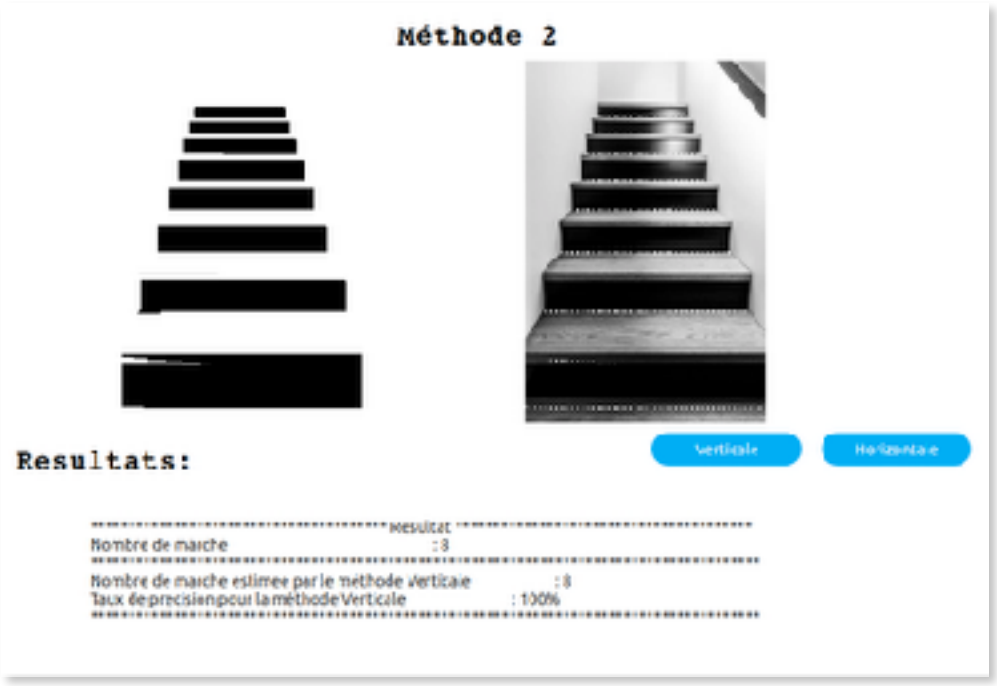

Image 31. Méthode 2. Balayage Vertical

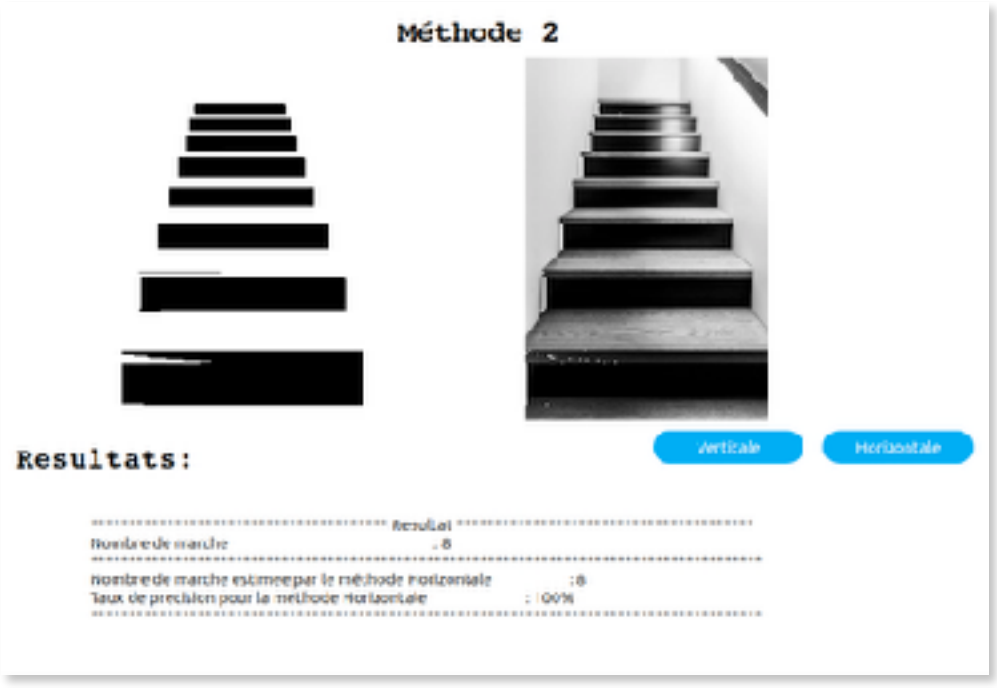

Image 32. Méthode 2. Balayage Horizontal

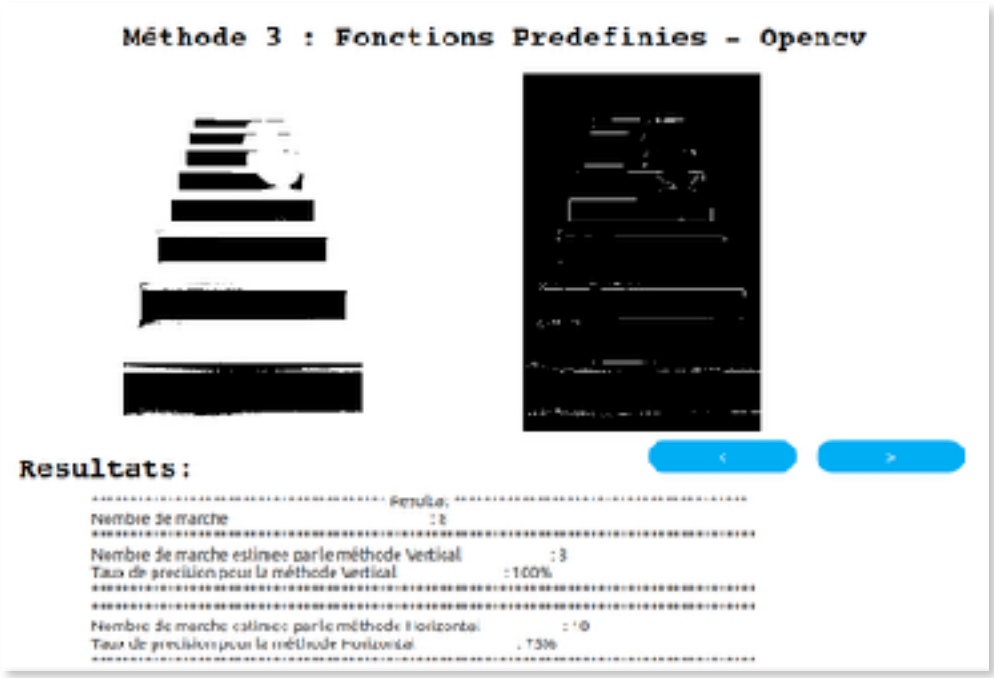

Image 33. Méthode 3 Threshold et Filtre de Canny

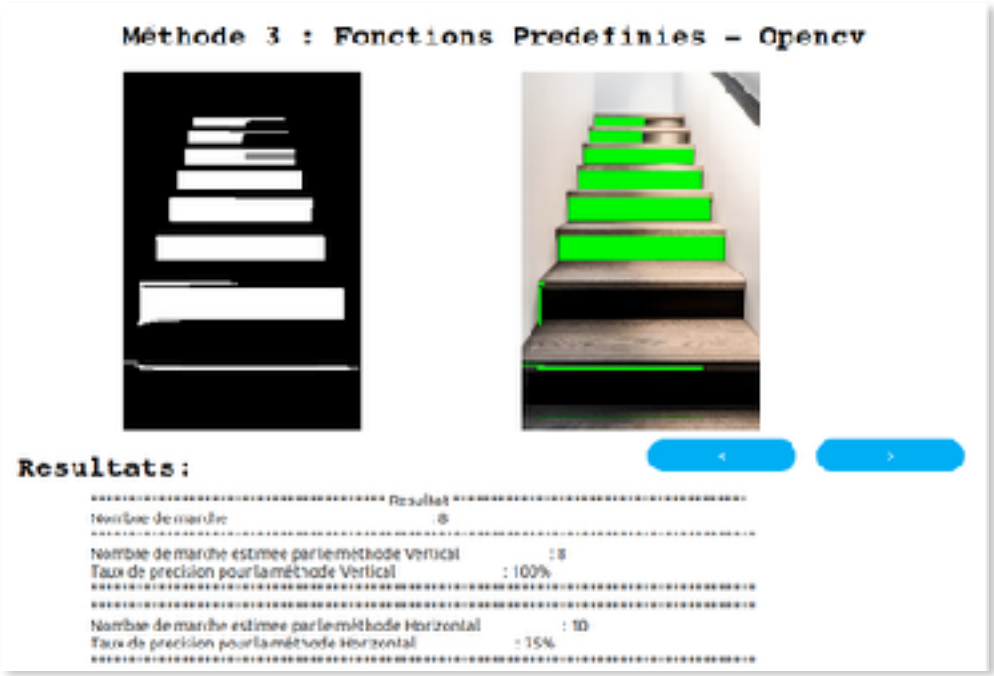

Image 34. Méthode 3. Détection des formes rectangulaires

Image 35. Méthode 3. Parcours Horizontal et Vertical