Détection des lignes aéroportuaires par méthode hybride de filtrage particulaire et de réseaux de neurones

Esteban Perrotin, Matthieu Roy, Ariane Herbulot, Michel Devy, Fabrice Bousquet

Détection des lignes aéroportuaires par méthode hybride de filtrage particulaire et de réseaux de neurones

Esteban Perrotin[12] Matthieu Roy[2] Ariane Herbulot[2]
Michel Devy [2] Fabrice Bousquet[1]

[1] AIRBUS Operations
[2] LAAS-CNRS

esteban.perrotin@airbus.com

Résumé

La vision par ordinateur est devenue un élément essentiel pour l'assistance à la conduite, notamment dans le contexte automobile avec des méthodes d'interprétation de scène de plus en plus poussées Cependant, les solutions existantes pour l'automobile poussent à l'utilisation de réseaux de neurones dits "end-to-end". Dans le contexte de l'aéronautique civile, ces solutions semblent difficiles à mettre en place, notamment à cause des aspects de certifications inhérentes au domaine applicatif. Dans cet article, nous décrivons le problème de détection de ligne sous la forme d'un suivi de trajectoire dans un contexte aéroportuaire. Pour répondre à ce problème, nous le découpons en sous-parties puis nous proposons une architecture modulaire pour y répondre. Parmi les techniques utilisées dans les modules de l'architecture, nous utilisons notamment un filtre à particule et un petit réseau de neurones convolutif pour de la classification binaire. L'intérêt principal de cet article est de fournir un exemple d'application de vision par ordinateur dans un cadre minimal pour l'aviation civile. L'objectif ultérieur sera de discuter des aspects de certification autour de cette fonction et des méthodes utilisées dans les différents modules. Nous présentons ici les détails de la méthode de détection de ligne et quelques résultats généraux obtenus dans les différents modules de l'architecture proposée.

Mots Clef

vision par ordinateur, détection de ligne, filtrage particulaire, réseaux de neurones convolutifs

Abstract

Computer vision has become an essential element for driving assistance, with a deep focus on scene interpretation. However, existing automotive solutions are pushing for the use of so-called "end-to-end" neural networks. In the context of civil aeronautics, these solutions seem difficult to implement, in particular because of the certification aspects. In this article, we describe the line detection problem in the form of trajectory tracking in an aeronautics context. To answer this problem, we cut it in sub-parts, and we propose a modular architecture to answer it. Among the techniques used in the modules of the architecture, we notably use a particle filter and a small convolutional neural network doing binary classification. The main interest of this article is to provide an example of a computer vision application in a minimal framework for civil aviation. The subsequent objective will be to discuss the certification aspects around this function and the methods used in the different modules. We present here the details of the line detection method and some general results obtained in the different modules of the proposed architecture.

Keywords

computer vision, line detection, particle filtering, convolutional neural network, certification

1 Introduction

La conduite assistée est un enjeu majeur du $21^{ème}$ siècle. Dans ce contexte, la vision par ordinateur a pris une place capitale dans la perception de l'environnement des véhicules. Parmi ses nombreux objectifs, on peut citer la détection des obstacles, la détection des éléments de signalisations ou encore l'estimation du comportement du véhicule au regard de son environnement (odométrie visuelle, SLAM, ...). Dans notre cas, nous nous intéressons à

Figure 1 – Input RGB image from the fin camera.

l'assistance à la navigation d'un aéronef sur les taxiways dans les zones aéroportuaires ; dans cet article, nous nous focalisons sur la détection de la signalisation à partir d'images acquises par des caméras visibles montées sur l'avion (Figure 1). Parmi les éléments de signalisation, les marquages au sol fournissent d'importantes informations pour la navigation. Ces informations pourront servir à construire diverses fonctions : garder une trajectoire fixe (maintien de l'axe), estimer la position relative et absolue de l'avion, suivre le tracé d'une ligne, ou encore alerter le pilote en cas de dérive afin de prévenir une potentielle sortie de piste. Dans nos travaux, nous cherchons à identifier une suite de points dans le plan de l'image (en pixels) qui décrivent des marquages. Nous proposons et développons des méthodes en tenant compte des aspects de certification propre à l'aviation civile. L'objectif à terme est d'ouvrir une discussion autours sur la certification de fonctionnalités aéronautiques utilisant la vision par ordinateur.

Dans cet article, nous présentons une méthode basée sur le filtrage particulaire afin d'extraire une chaine de sous fenêtres (imagettes) à partir de l'image décrivant une ligne. Les autres traitements (extraire l'équation de ligne, identifier le type du marquage...) ne sont pas évoqués dans la suite de l'article.

1.1 Travaux relatifs

Dans le domaine automobile, la détection de lignes est un sujet bien étudié depuis la fin des années 90. L'objectif est généralement de détecter les différentes voies de circulation. Parmi les solutions proposées, un nombre important de méthodes de détection reposent sur l'utilisation de la transformation de Hough afin d'extraire les lignes et les courbes [1] dans l'image à partir des contrastes entre la route et le marquage. La détection de ces voies a aussi été réalisée par plusieurs méthodes de filtrage, notamment en utilisant le filtrage de Kalman [2]. Le filtrage s'effectue au cours du temps. Plus récemment, les méthodes par réseaux de neurones ont permis une avancée dans le domaine [3]. Ces différentes méthodes ont été essayées et comparées dans l'état de l'art pour le domaine automobile [4, 5]. Elles reposent souvent sur une architecture "end-to-end". Le réseau est alors considéré comme une boite blanche difficilement explicable.
Malgré les bonnes performances des réseaux de neurones, leurs utilisations dans le contexte aéroportuaire est limité notamment dû aux contraintes du processus de certification des fonctions difficilement réalisable pour des réseaux de neurones dit "end-to-end". Dans cet article notre objectif est de détecter principalement les marquages au sol définissant les trajectoires à suivre et les délimitations des voies de circulation des avions. On propose d'utiliser une architecture modulaire, utilisant des éléments dont l'utilisation n'est pas encore certifiée (en particulier le filtrage particulaire et les réseaux de neurones à convolution). L'intérêt principal est de fournir ici un contexte minimal pour faciliter l'étude de la certification de ces méthodes.

Pour détecter les marquages aéroportuaires, une précédente version avait été proposée [6]. Cette approche utilise aussi le principe du filtrage particulaire. Cependant, le modèle dynamique est plus simple et considère uniquement la propagation selon l'axe vertical de l'image. Il faut alors transposer l'image pour obtenir les marquages horizontaux puis fusionner les données. De plus, la sélection des différentes particules se fait selon deux informations basées sur la couleur (pour les marquages jaunes) et sur le gradient entre tarmac et marquage. Dans notre cas, nous considérons un modèle plus complexe qui capture directement la géométrie des lignes. La sélection des particules se fait à partir d'un classifieur d'image qui prédit la probabilité que le voisinage d'une particule corresponde à un marquage. Notre modèle se rapproche aussi des travaux de Tran et al. [7] et des travaux de Prateek et al. [8]. Dans leurs travaux, Tran et al. traquent un objet au cours du temps. L'estimation du déplacement au cours du temps se fait par filtrage particulaire. La sélection des particules se fait par la transformation de Hough généralisée dont la référence est mise à jour à chaque itération.

1.2 Organisation de l'article

Dans la section suivante nous modélisons le problème de détection des lignes comme un modèle markovien et présentons l'architecture générale de détection. Dans la section 3, nous détaillons l'algorithme de filtrage particulaire. La section 4 présente succinctement les autres parties de l'architecture et présente la construction du réseau de neurones. Puis la section 5 présente les résultats obtenus. Finalement, nous concluons sur l'intérêt de cette méthode, les prochaines évolutions et nos contributions.

2 Modélisation du problème et architecture de la solution

2.1 Vision Markovienne

Pour représenter ces lignes, nous les considérons comme une succession de points liés par une chaine de Markov. Chacun de ces points peut se représenter sous la forme d'un vecteur d'état $X_k = (x_k, y_k, \theta_k, s_k)$, avec (x_k, y_k) représentant les coordonnées du point dans le plan de l'image, θ_k l'orientation du marquage dans l'image en ce point et s_k la taille région d'intérêt considérée autour de ce point. À partir de ces paramètres, on peut décrire une imagette $\mathcal{N}_k$ centrée en (x_k, y_k) de taille $s_k \times s_k$ px. Cette imagette contient le motif de la ligne. Figure 2 montre une évolution possible de ces imagettes selon une succession d'états. L'ensemble $\{X_k\}_{k=0}^{p}$ décrit la ligne. Notre objectif est d'estimer cette succession d'états à partir d'un état initial X_0 de sorte à capturer l'information géométrique de la ligne. Pour ce faire, nous décrivons un modèle dynamique modélisant le passage d'un état X_k à l'état suivant X_{k+1}.

2.2 Modèle d'évolution spatiale des motifs

En posant ce problème sous la forme d'une chaîne de Markov, il s'apparente à un problème de suivi de trajectoire. Ce type de problème admet plusieurs solutions étudiées [9] selon le modèle dynamique considéré de la trajectoire et de l'apriori entre les mesures et l'évolution du système. Le modèle dynamique décrit l'évolution des paramètres de l'espace d'état à chaque itération. On suppose qu'il existe une fonction g qui décrit la transition du système de l'itération k à l'itération $k+1$:

$$X_{k+1} = g(X_k) \tag{1}$$

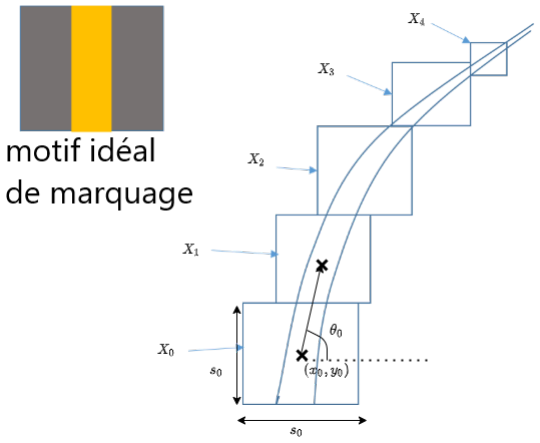

Figure 2 – Principe de détection de ligne selon un modèle Markovien. La succession des états X_k décrit la progression de la ligne à travers l'image

En pratique l'évolution entre les états n'est pas connue. On choisit d'approximer la fonction g par le modèle $\mathcal{M}$ en supposant l'évolution des paramètres comme suit :

$$x_{k+1} = x_k + \frac{1}{2} s_k cos(\theta_k) \tag{2a}$$

$$y_{k+1} = y_k + \frac{1}{2} s_k sin(\theta_k) \tag{2b}$$

$$\theta_{k+1} = \theta_k \tag{2c}$$

$$s_{k+1} = f(x_k, y_k) \tag{2d}$$

Le lien entre deux états est considéré par la relation :

$$X_{k+1} = \mathcal{M}(X_k, U_k) \tag{3}$$

Il s'agit d'un modèle simple qui ne capture qu'une petite partie de la géométrie des lignes. Avec ce modèle les lignes sont considérées comme une succession de segments suivant une trajectoire rectiligne uniforme. Cette approximation entraîne une erreur potentielle entre le modèle et la réalité observée. Cette erreur est considérée comme un bruit présent par le terme U_k. Ce bruit n'est pas a priori un simple bruit blanc Gaussien. Il serait possible d'ajouter un terme supplémentaire qui capture plus précisément la géométrie des lignes. Dans notre cas, ce modèle est suffisant pour fournir des résultats satisfaisants. La taille de la région d'intérêt s_k est estimée par la fonction f. Cette fonction estime le nombre de pixels décrivant une ligne de 15cm d'épaisseur sur un sol plat à partir des paramètres de la caméra et du point (x_k, y_k) dans l'image. Cette fonction est basée sur une approche géométrique et n'est pas détaillée ici. La taille de la région d'intérêt sert aussi à fixer la "vitesse" d'avancement des imagettes. L'intérêt

principal est de réduire la distance entre les points décrivant la ligne, lorsque ceux-ci s'éloignent de la caméra ; L'éloignement induit plus d'artefacts et complexifie la détection. Un deuxième intérêt et de conserver la proportion entre le nombre de pixels définissant le marquage et le nombre de pixels définissant le tarmac.
Notre problème principal revient à estimer X_{k+1} à partir de X_k. Malheureusement, à cause du terme de bruit U_k nous ne connaissons pas la vraie valeur de X_{k+1}. Nous disposons seulement d'approximations notées $\hat{X}_{k+1}$.

2.3 Modèle d'observation

Pour répondre à ce problème, nous construisons un modèle $\mathcal{H}$ qui attribue un score de vraisemblance à un vecteur X_k. Ce score est noté Z_k. Il est attribué en fonction des paramètres qui composent le vecteur d'état et de l'image. Cette fonction traduit la probabilité que l'état X_k correspondent à l'état recherché X_k vis à vis de la mesure. Dans notre cas, la mesure (noté Z) est l'image entière.

$$Z_k = \mathcal{H}_Z(X_k, V_k) \tag{4}$$

Si les paramètres du vecteur X_k décrivent une imagette $\mathcal{N}_k$ qui correspond à un motif de marquage désiré et si certaines caractéristiques géométriques de cette imagette correspondent à celle dans le vecteur d'état, alors le score qui lui est attribué sera élevée (proche de 1) sinon ce score sera faible (proche de 0). De plus, l'attribution du score peut être erronée. Un terme de bruit V_k suivant une distribution non connue modélise ces erreurs potentielles

En résumé, nous désirons approximer la suite des $\{X_k\}_{k=0}^{p}$ à partir d'un état initial X_0, d'un modèle dynamique $\mathcal{M}$, d'une mesure Z_k associée à X_k par un modèle $\mathcal{H}$. Il faut aussi estimer un état initial X_0 et détecter un état final X_p. Pour résoudre ces problèmes, on présente dans la partie suivante notre architecture. Puis, dans les sections suivantes, nous détaillons chacune de ces parties.

2.4 Architecture modulaire pour la détection de lignes

Pour répondre à notre problématique, nous proposons de séparer chacune des parties du problème en différentes sous-parties traitées par des modules indépendants. Le premier module est celui d'initialisation. Il a pour but de proposer un état initial X_0. Le second module est celui dit d'observation. À partir d'un vecteur d'état

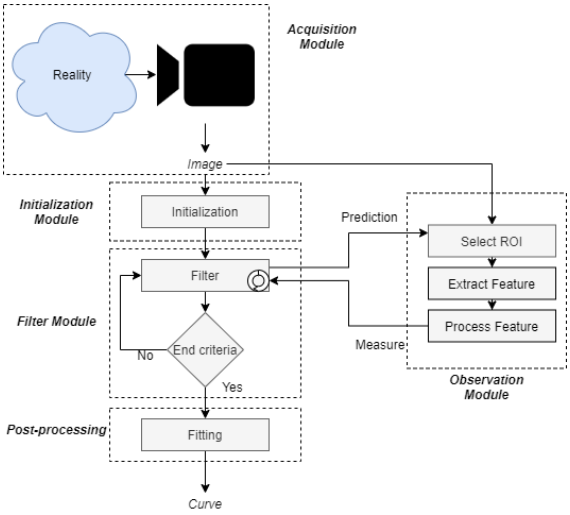

Figure 3 – Schéma simplifié de notre architecture modulaire de détection de ligne aéroportuaire.

X_k, il construit l'imagette $\mathcal{N}_k$, extrait différentes propriétés de cette imagette et associe un score Z_k . Ce score correspond à la vraisemblance que X_k corresponde à un marquage correspondant vis-à-vis de l'image. Ce module correspond au modèle $\mathcal{H}$. Enfin, un module filtre le vecteur d'état courant et prédit un nouveau vecteur X_{k+1} à partir du modèle $\mathcal{M}$, des vecteurs précédents $X_{0:k}$ et des mesures obtenues $Z_{0:k}$ tant qu'un critère d'arrêt n'est pas respecté. L'agencement entre les modules est montré Figure 3. Il est aussi possible d'ajouter des traitements a posteriori, comme par exemple estimer une équation de ligne à partir des $(x_{0:k}, y_{0:k})$.

L'intérêt principal de cette architecture est de séparer les parties qui utilisent l'image entièrement, légèrement ou aucunement. En effet, l'initialisation utilise l'image entièrement. L'observation utilise seulement l'imagette $\mathcal{N}_k$ et le filtre n'utilise pas directement l'image. Dans le contexte de l'aéronautique civile, chaque fonction et ses sous-parties doivent pouvoir être décrites par des exigences compréhensibles et vérifiables. Cette décomposition a pour but d'aider cet aspect. Un autre avantage de cette modularité est la possibilité de changer et d'améliorer chaque module sans toucher aux autres.

3 Estimation de la ligne par filtrage particulaire

On souhaite estimer la séquence des $X = \{X_k\}_{k=0}^{p}$ la plus probable au vue de l'image. Pour ce faire, deux estimateurs sont généralement uti-

lisés, l'erreur quadratique minimale :

$$\hat{X}_{MMSE} = \int_X XP(X|Z)\mathrm{d}X \tag{5}$$

Et l'estimateur du maximum a posteriori :

$$\hat{X}_{Map} = \underset{X}{argmax}\{P(X|Z)\} \tag{6}$$

Ces deux quantités demandent de connaître la densité de probabilité $p(X|Z)$. Malheureusement, il n'est pas envisageable de générer toutes les réalisations de X. Cette densité de probabilité est inconnue. On utilise une méthode de filtrage particulaire qui permet d'approximer cette densité par une suite discrète de variables aléatoires pondérées.

3.1 Principe de base du filtrage particulaire

Le filtrage particulaire est une méthode de simulation séquentielle de type Monte-Carlo. Les particules explorent l'espace d'état en évoluant selon un modèle dynamique. Un processus de sélection exploitant des mesures vient concentrer les particules dans les régions d'intérêts de l'espace d'état. La figure 4 montre les différentes étapes du filtre à particule.

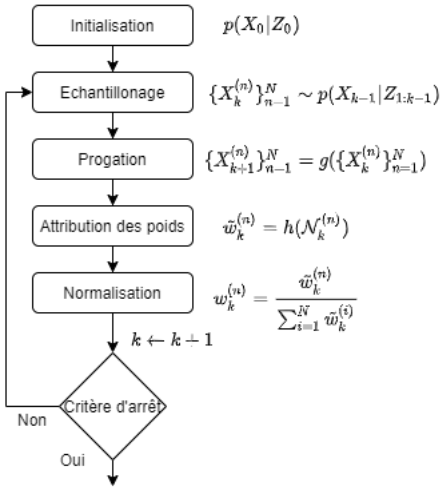

Figure 4 – Schéma général du filtre particulaire.

Les poids sont attribués par rapport à la vraisemblance d'obtenir le score Z_k en connaissant la réalisation du vecteur d'état X_k^n d'une particule, ainsi que la probabilité de transition d'un état X_k vers un état X_{k+1} :

$$w_k^n = w_{k-1}^n \frac{P(Z_K|X_k^n)P(X_k^n|X_{k-1}^n)}{Q(X_k^n|X_{k-1}, Z_{0:k})} \tag{7}$$

Les particules sont tirées selon une loi a priori Q :

$$X_k^{(n)} \sim Q(X_k|X_{k-1}^{(n)}, Z_{0:k}) \tag{8}$$

3.2 Mise à jours des poids

Dans un premier temps, nous décidé de tirer les particules selon la loi de transition a priori du système, de sorte que :

$$P(X_k^n|X_{k-1}^n) \sim Q(X_k^n|X_{k-1}, Z_{0:k}) \tag{9}$$

La distribution choisie est une gaussienne centré. La mise à jours des poids devient :

$$w_k^n = w_{k-1}^n P(Z_K|X_k^n) \tag{10}$$

Dans un deuxième temps, on propose de mettre les poids en tenant compte du score obtenu par chacune des particules ainsi que la distance entre elles dans l'espace d'état.

$$L_k(X_k^n) = \sum_{i=0}^{N} e^{-\frac{d(X_k^n; X_k^i)}{(z_k^i)^2}} \tag{11}$$

La fonction d définit la distance entre de deux états. Nous choisissons ici d'utiliser la distance entre les coordonnées des vecteurs dans l'image :

$$d(X_k^i; X_k^j) = \|(x_k^i, y_k^i) - (x_k^j, y_k^j)\|_2 \tag{12}$$

La mise à jours des poids est alors définie par :

$$w_k^n = w_{k-1}^n P(Z_K|X_k^n) L_k(X_k^n) \tag{13}$$

Cette formule de mise à jours des poids permet de favoriser les regroupements de particules dont les paramètres spatiaux sont proches. En pratique, cela diminue le risque d'avoir une particule qui s'éloigne trop de la masse et attire les autres à elles.

3.3 Algorithme du filtre

Le pseudo-algorithme 1 décrit les principales étapes utilisées pour le filtrage dans notre application. Les poids sont mis à jours selon l'équation 10 ou 13. L'échantillonnage est fait selon une méthode plus détaillée dans [6].

Algorithm 1 (Un cycle) Calculer une approximation de la densité $P(X_k|Z)$ par l'ensemble $\{X_k^{(n)}, w_k^{(n)}\}_{n=1}^N$ en associant à chaque particule X_k^n un poids w_k^n à partir d'une distribution initiale $\{X_0^{(n)}, w_0^{(n)}\}_{n=1}^N$

(Prédiction) Construire X_{k+1}^n à partir de X_k^n en utilisant l'équation 2

(Obtention des mesures) Calcul de la vraisemblance $P(Z|X_k^n)$ des particules vis à vis de l'image à partir du modèle d'observation $\mathcal{H}$ (équation 4)

(Attribution des poids) Puis, mettre à jour les poids :

$$\tilde{w}_k^n = w_{k-1}^n L_k(X_k^n) Z_k^n \tag{14}$$

(Normalisation des poids) Normaliser les poids des particules, de sorte que leur somme soit égale à un : $w_k^n = \frac{\tilde{w}_k^n}{\sum_{i=1}^N \tilde{w}_k^i}$

(Échantillonnage) Tirer un nouvel échantillon de N particules selon la distribution $\{X_k^{(n)}, w_k^{(n)}\}_{n=1}^N$

Le détail de l'attribution du score de vraisemblance est donnée dans la section 4.

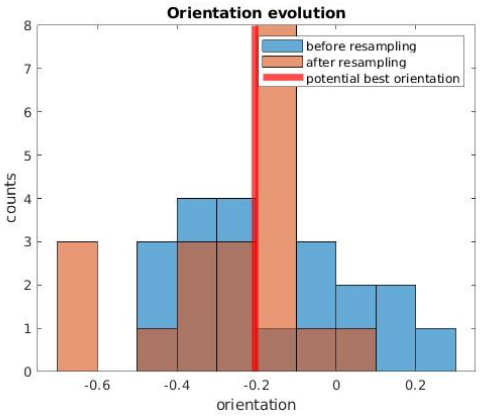

Figure 5 – Histogrammes des orientations avant et après le rééchantillonnage. La ligne verticale en rouge représente l'orientation la plus pertinente créée à partir des données simulées.

3.4 Vérification du modèle

Pour vérifier que le bon fonctionnement du filtre et du avec le modèle dynamique, nous simulons des lignes à partir de courbes de Bézier d'ordre 5. La mesure attribuée aux vecteurs d'état est construite en utilisant la vérité terrain. Pour ce faire, on calcule la distance entre la particule et le plus le plus proche sur la courbe. On associe également l'orientation idéale à partir de ce point. Si cette distance est faible et que l'orientation proposée par la particule correspond à l'orientation observée au point le plus proche de la courbe alors le poids attribué à la particule sera élevé. Après rééchantillonnage les particules ont plus de chances d'avoir une orientation pertinente et d'être plus proche de la courbe. La figure 5 montre l'histogramme des orientations des particules avant rééchantillonnage (en bleu) et après rééchantillonnage (en orange). La ligne rouge correspond à l'orientation idéale moyenne aux points de la courbe les plus proches des particules. La Figure 6 montre un exemple de suivi de ligne notre filtrage particulaire.

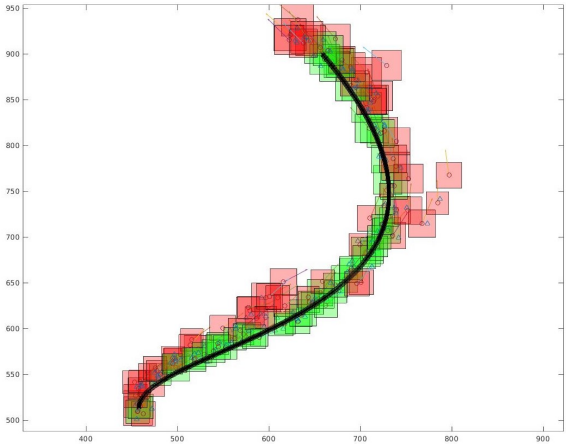

Figure 6 – Exemple de test pour vérifier la solution par filtrage particulaire avec le modèle dynamique proposé. En rouge les particules de poids faibles, en vert les particules de poids forts.

4 Attribution du score par réseaux de neurones et méthodes géométriques

4.1 Initialisation

L'initialisation a pour but de proposer un état X_0 pertinent. Pour ce faire, nous commençons par considérer différentes régions d'intérêts. Ces régions dépendent de la position de la caméra et viennent masquer principalement l'avion et le ciel. Ensuite, à chaque pixel est attribué une distance à une couleur de référence (généralement jaunes pour les lignes du taxiway). Puis, on applique différents opérateurs morphologiques pour accentuer localement le contraste entre les éléments jaunes et les éléments gris. Cette étape a pour but de souligner les gradients correspondant très probablement à une interface entre le tarmac et les marquages. On segmente ensuite les pixels selon un seuil calculé par la méthode de Ot'su. A cette étape, la majorité des éléments extraits correspondent à des morceaux de marquages suffisamment résolus avec une peinture visible. Finalement, un algorithme de squelettisation permet d'extraire des points deux à deux de ces éléments. Le vecteur X_0 est alors construit par la position d'un point, par l'orientation de ce point vis à vis du squelette et une vitesse attribuée selon sa distance à la caméra (toujours avec l'hypothèse que le sol est plat).

4.2 Construction de mesures associées aux vecteurs d'états

Pour pouvoir résoudre notre problème 1, nous avons besoin d'associer à tout vecteur d'état X_k une valeur correspondant à la probabilité que les paramètres de ce vecteur décrivent un marquage. Pour ce faire, nous construisons trois fonctions qui évaluent si l'imagette $\mathcal{N}_k$ contient un élément principal qui est centré dans l'imagette, dont l'orientation est proche du paramètre θ_k et qui est reconnu par un classifieur comme étant un marquage. Ces fonctions attribuent un score entre 0 et 1. La valeur finale renvoyée est le produit de ces scores.

Mesure de l'orientation et l'alignement. Les valeurs attribuées pour l'orientation et l'alignement sont calculées à partir d'étapes similaires à celles utilisées dans l'initialisation. On effectue une segmentation colorimétrique et une squelettisation selon les mêmes étapes que précédemment. Puis on effectue une transformation de Hough pour extraire la droite principale du squelette. À partir des paramètres de cette droite dans l'espace de Hough on obtient directement son orientation. Il reste simplement à calculer la distance du centre de l'imagette à son projeté sur la droite et à normaliser cette valeur pour obtenir le score désiré.

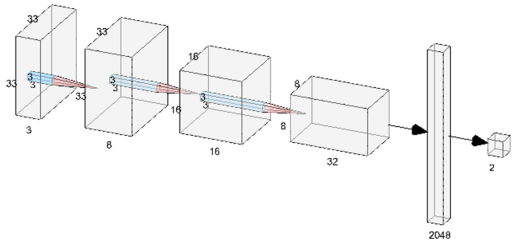

Figure 7 – Architecture du réseau de neurones utilisé. Il s'agit d'une succession de couche de convolution et de pooling relié à une couche complètement connectée.

Réseaux de neurones convolutifs. Le classifieur choisi dans cet article est un réseau de neurones à convolution (CNN). Ces dernières années, les réseaux de neurones ont percé dans l'état de l'art dans de nombreux domaines. C'est en particulier le cas dans la classification d'images. Nous avons donc implémenté une architecture assez simple de CNN pour répondre à notre besoin mais en restant dans un contexte très simple.
L'architecture du réseau est montrée sur la figure 7. L'imagette de taille $s_k \times s_k$px est redimensionnée en 33×33px puis est donnée en entrée du réseau. Une succession de plusieurs couches de convolutions et de pooling sont appliquées pour extraire des éléments descriptifs. La dernière couche est une couche totalement connectée qui produit en sortie un résultat entre 0 et 1 correspondant à la confiance sur la présence d'un marquage dans l'imagette. Nous avons aussi fait plusieurs essais sur l'importance de la normalisation des données. Il est apparu que les résultats sont meilleurs en normalisant les données à la sortie des couches de convolution.

Construction du jeu de données d'apprentissage. Le jeu de données est composé d'imagettes de taille 33×33 pixels sélectionnées à partir d'images annotées manuellement. On tire au hasard 2000 imagettes à partir de plusieurs images selon une probabilité uniforme. Si au moins 100 pixels de l'imagette ont été annotées comme appartenant à un marquage, alors on classe l'imagette dans la catégorie 'marquage'. Sinon, l'imagette est classée dans la catégorie 'non marquage'. La figure 8 montre des exemples d'imagettes.
Cette méthode de construction du jeu de données

a l'avantage d'être rapide et facile à implémenter mais comporte deux inconvénients. L'annotation manuelle des pixels n'est pas parfaite, ce qui peut apporter des faux positifs dans la base de données. De plus, le seuil fixé à 100 pixels est arbitraire. Si une imagette contient 99 pixels manuellement annotées comme faisant partie de la classe 'marquage', l'imagette sera notée dans la classe 'non-marquage'. Ce qui peut être considéré comme un faux négatif. Les performances des algorithmes de classification sont donc à relativiser vis-à-vis de la précision de la base de données.

Figure 8 – La première ligne montre des imagettes considérées comme positives. La seconde ligne montre des imagettes considérées négatives.

5 Résultats

Nous présentons ici des résultats qualitatifs préliminaires. Des résultats quantitatifs sont en cours de production. La figure 9 présente les résultats actuels sur des images de basses qualitées en 1280x800 pixels. Le nombre de points proposés par l'initialisation et la sensibilité du critère d'arrêt complique la détection correcte de toutes les lignes.
La méthode est encore en cours d'amélioration. Parmi les améliorations en cours, le suivi temporel permettra d'améliorer la robustesse des points d'initialisation et d'obtenir une meilleure approximation des courbes. De plus, de nouvelles initialisations seront proposées à chaque croisement de lignes. Plusieurs paramètres sont en cours d'amélioration pour affiner les détections lointaines et avoir plus de justesse vis-à-vis du critère d'arrêt. Un des avantages de la méthode proposée est de pouvoir améliorer indépendamment chaque modules au fur à mesure.

6 Conclusion et perspectives

Dans cet article, nous proposons une méthode de détection de ligne aéroportuaire à partir d'une architecture modulaire. Cette architecture est pertinente grâce à la découpe du problème de base en différentes sous parties. Une ligne est considérée comme l'évolution d'un vecteur d'état dont la solution peut se calculer à partir de méthodes de filtrage classique. Au vu du modèle posé, nous avons choisi l'utilisation d'un filtre particulaire qui se prête bien à ce type de problème. Pour attribuer les poids des particules du filtre, nous utilisons une méthode hybride reposant sur l'extraction de caractéristiques géométriques et d'un réseau de neurones convolutif classifiant la présence ou l'absence du motif de la ligne. Nous présentons deux façons de mettre à jour les poids des particules du filtre. Nous pensons que ce type de méthode hybride pourrait être utilisé dans plusieurs autres applications : le suivi temporel d'objets, la fusion de données, etc. Dans le contexte aéronautique, cette application fournit un cadre concret et intéressant pour travailler à la certification de différentes méthodes dont les réseaux de neurones à convolution et le filtrage particulaire. Ces aspects seront étudiés dans de futurs travaux.

Figure 9 – Résultats obtenus avec notre méthode. Les croix rouges sont les points proposés par l'initialisation. Les points bleus correspondent au maximum de vraisemblance des particules. Les équations des courbes sont en rouges. Un seuillage local est appliquée pour détecter quels pixels appartiennent aux marquages.

Remerciements

Cette étude a été réalisée en collaboration avec Airbus Operations et le LAAS-CNRS.

Références

[1] A. Kumar and P. Simon, "Review of lane detection and tracking algorithms in advanced driver assistance system," *International Journal of Computer Science and Information Technology*, vol. 7, pp. 65–78, 08 2015.

[2] R. Aufrer, F. Marmoiton, R. Chapuis, F. Collange, and J. Derutin, "Détection de route et suivi de véhicules par vision pour l'acc," *GRETSI, Saint Martin d'Hères, France*, 2000.

[3] M. Haris and A. Glowacz, "Lane line detection based on object feature distillation," *Electronics*, vol. 10, no. 9, 2021.

[4] A. Kumar and P. Simon, "Review of lane detection and tracking algorithms in advanced driver assistance system," *International Journal of Computer Science and Information Technology*, vol. 7, pp. 65–78, 08 2015.

[5] J. Cao, S. Song, W. Xiao, and Z. Peng, "Lane detection algorithm for intelligent vehicles in complex road conditions and dynamic environments," *Sensors*, vol. 19, p. 3166, 07 2019.

[6] C. Meymandi-Nejad, S. E. Kaddaoui, M. Devy, and A. Herbulot, "Lane detection and scene interpretation by particle filter in airport areas," in *14th International Conference on Computer Vision Theory and Applications (VISAPP 2019)*, (Prague, Czech Republic), Feb. 2019.

[7] A. Tran and A. Manzanera, "A versatile object tracking algorithm combining particle filter and generalised hough transform," in *2015 International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp. 105–110, 2015.

[8] P. K. Gaddigoudar, T. R. Balihalli, S. S. Ijantkar, N. C. Iyer, and S. Maralappanavar, "Pedestrian detection and tracking using particle filtering," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 110–115, 2017.

[9] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Trans. Signal Process.*, vol. 50, pp. 425–437, Feb. 2002.