Ecole Nationale des
Sciences Géographiques

SNCF Réseau

Rapport de stage

Cycle des Ingénieurs diplômés de l'ENSG 3ème année

Détection des rails sur des images à partir de caméras d'environnement

Tasnîme-Jenna Louartani

Mars-Septembre 2023

☒ Non confidentiel ☐ Confidentiel IGN ☐ Confidentiel Industrie ☐ Jusqu'au ...

Ecole Nationale des Sciences Géographiques
6-8 Avenue Blaise Pascal - Cité Descartes - 77420 Champs-sur-Marne
Téléphone 01 64 15 31 00 Télécopie 01 64 15 31 07

Jury

Président de jury :

Commanditaire :

Encadrement de stage :

Kader Benabdeli

Enseignant référent :

Malika Grim-Yefsah

Rapporteur expert :

Responsable pédagogique du cycle Ingénieur :

Jean-François Hangouët, IGN/ENSG/PEGI

Gestion du stage :

Delphine Genes, Claire Driessens, IGN/ENSG/DFI

Stage de fin d'étude du xxx au xxx

Diffusion web : ☒ Internet ☒ Intranet

Situation du document :

Rapport de stage de fin d'études présenté en fin de 3ème année du cycle des Ingénieurs

Nombres de pages : 61 pages dont 7 d'annexes

Système hôte : LaTeX

Modifications :

EDITION	REVISION	DATE	PAGES MODIFIEES
1	0	09/2016	Création

Remerciements

Je tiens à adresser mes plus sincères remerciements à toutes les personnes qui ont contribué au bon déroulement de mon alternance et de mon stage.
Tout d'abord, M. Kader Benabdeli, mon tuteur de stage et responsable de la GAIA FAB et de l'équipe I2D pour son soutien, ses conseils et pour le temps consacré à mon apprentisage. Les compétences et les connaissances que j'ai acquis dans cette équipe vont me permettre d'étoffer mon parcours professionnel.
Les membres de l'équipe I2D qui m'ont accompagnés avec bienveillance et m'ont transmis leur expertise.
Mon école, l'ENSG-Géomatique, plus particulièrement Mme Malika Grim et M. Sébastien Mustière, les responsables de la filière DesiGeo, pour avoir cru en mon potentiel.
Enfin, tous mes collègues de travail pour leur disponibilité.
J'ai également une pensée chaleureuse pour mes proches, famille et amis, qui n'ont cessé de me soutenir et de m'encourager au quotidien dans mes démarches, à la fois sur le plan professionnel et personnel.
Un dernier remerciement, enfin, à Ludovic Gerbet, pour le temps consacré à la relecture et à la correction de ce rapport...

Abstract

Afin de répondre aux besoins de SNCF Réseau, le département GAIA FAB est chargé de digitaliser l'infrastructure du réseau national français. Après un travail de collecte de divers documents techniques décrivant le réseau ferré, la GAIA FAB saisit manuellement les informations dans un outil appelé NetGeo qui alimentent la base GAIA. Diverses erreurs proviennent lors de la saisie : il y a donc des différences entre la réalité terrain et la base de données. Afin de redresser la donnée, des images provenant de caméras embarquées sur les engins de surveillance des voies sont utilisées pour détecter les objets de l'infrastructure ferroviaire de manière la plus automatisée possible. Le but étant de déterminer la position des objets tant sur le référentiel géographique que le référentiel ferroviaire. Pour cela, il faut savoir à quelle voie un objet est rattaché. Ainsi, la détection des voies sur les images apporte une aide au niveau du redressement de la donnée. A partir de deux approches, la finalité est de déterminer les rails sur les images provenant de caméras embarquées. Il y a tout d'abord un travail de recherche autour du Deep Learning révelant les limites de cette technologie dans le cadre de cet exercice et ensuite un travail utilisant la morphologie mathématique des images. La deuxième approche permet de situer la voie sur laquelle le train circule et ses voies parallèles en faisant des analyses colorimétriques.

Mots clés : Deep Learning, rails, voies, ESV, Hough, Canny, morphologie mathématique

Abstract

To meet the needs of SNCF Réseau, the GAIA FAB department is responsible for digitizing the infrastructure of the French national network. After collecting various technical documents describing the rail network, GAIA FAB manually enters the information into a tool called NetGeo, which feeds the GAIA database. Various errors occur during data entry: there are therefore differences between the reality in the field and the database. In order to rectify the data, images from on-board cameras on track surveillance vehicles are used to detect objects on the railway infrastructure as automatically as possible. The aim is to determine the position of objects in both geographical and railway frames of reference. To achieve this, it is necessary to know to which track each object is attached. To this end, the detection of tracks in the images helps to rectify the data. Two approaches are used to determine the tracks on images from on-board cameras. The first is a research project based on Deep Learning, revealing the limits of this technology in the context of this exercise, and the second is a project using mathematical image morphology. The second approach uses colorimetric analysis to locate the track on which the train is travelling and its parallel tracks.

Key words: Deep Learning, railway, Hough, Canny, Mathematical morphology

Table des matières

Glossaire et sigles utiles

ENSG École Nationale des Sciences Géographiques

SNCF Société Nationale des Chemins Ferrés

GPS Global Positionning System

DGII Direction Générale Industrielle et ingénierie

IP3M Intégration Projets Multi Métiers

DS Données et Services

I2D Ingénierie du Data et du Digital

ESV Engins de Surveillance des Voies

PK Point Kilométrique

TIV Tronçon Itinéraire de Voie

SIG Système d'Information Géographique

GeoTIV Tronçon Itinéraire de Voie avec la position géographique

SAM Segmentation Anything Model

ROI Region of Interest, Région d'intérêt

Introduction

0.0.1 Présentation de l'entreprise

SNCF Réseau est une branche de la Société Nationale des Chemins Ferrés (SNCF) qui s'occupe d'assurer la sécurité, l'entretien, la modernisation et le développement du réseau ferroviaire. Au sein de SNCF Réseau cohabitent plusieurs directions, notamment la Direction Générale Industrielle et ingénierie (DGII) qui conçoit les ouvrages et les installations ferroviaires. Elle comprend la division Intégration Projets Multi Métiers (IP3M) qui développe une ingénierie de l'intégration en lien avec l'exploitation ferroviaire et la maintenance système. Le Pôle Données et Services (DS) s'occupe de décrire le réseau et de numériser le patrimoine. Dans ce pôle, il y a la Filière Description du Réseau que j'ai pu intégrer en septembre 2022. Dans le cadre d'une démarche de recherche d'entreprise, j'ai rejoint l'équipe de Kader Benabdeli à l'Ingénierie du Data et du Digital (I2D) qui travaille sur l'analyse des besoins et de l'environnement et valorise la donnée en utilisant différents référentiels. I2D travaille en étroite collaboration avec la GAIA FAB qui collecte, maintien et fournit les données descriptives du réseau.

0.0.2 Contexte

En effet, la GAIA FAB produit les données à partir de saisie manuelle des informations des plans techniques de l'infrastructure ferroviaire (notamment les pancartes de la signalisation ferroviaire, leurs implantations ou encore leurs équipements au sol). Ces données sont répertoriées dans la base de données **GAIA**. Diverses erreurs telles que les données sources obsolètes ou des erreurs dans la saisie manuelle sont présentes et donc il y a des différences entre la base de données GAIA et la réalité terrain. En parallèle, l'équipe I2D consomme les données GAIA pour répondre aux enjeux métiers et créer des solutions logicielles. Alors, I2D n'est plus seulement consommateur de ces données mais aide aussi à les corriger. Pour établir une comparaison entre les données de GAIA et la réalité terrain, I2D utilise une nouvelle ressource de données : les images, à partir de caméras embarquées à l'avant et l'arrière d'engins de maintenance, qui se nomment les ESV. Ainsi, une solution informatique et la plus automatisée possible est mise en place : la détection d'objets à partir des images des caméras embarquées en utilisant le Deep Learning. Une fois les objets détectés, il faut les affilier à la bonne voie. Par la suite, il est possible de comparer et corriger les données obtenues et celles saisies dans GAIA : c'est le croisement des données.

0.0.3 Problématique

Comme expliqué dans le contexte, la correction s'établit après le croisement des données entre GAIA et la base issue du terrain. Avec la complexité du réseau, il est possible d'avoir des cas où plusieurs objets identiques sont sur la même image et doivent être correctement affiliés aux voies. C'est là que la détection des voies intervient afin de pouvoir faire le croisement des données. Par conséquent, la

détection des objets dépend de la détection des voies sur l'image car c'est grâce à cette étape que le croisement des données peut se faire. C'est le premier enjeu de la détection des voies.

Le deuxième enjeu concerne le changement de référentiel. Historiquement, la SNCF a développé un référentiel propre aux métiers du ferroviaire. Il se repose sur trois principales mesures : la ligne, la voie et le point kilométrique (PK). D'un côté, les objets dans GAIA sont inscrits dans ce référentiel car la GAIA FAB se sert de documents ferroviaires techniques pour les administrer. D'un autre côté, les images récupérées des ESV sont géolocalisées (en Lambert93 ou en WGS84) donc les objets sont détectés dans le référentiel géographique. Afin de pouvoir croiser les données, il faut que les objets soient sur le même référentiel. Ainsi, les objets détectés automatiquement passent du référentiel géographique au ferroviaire. Pour se faire, on utilise les processus de la Ferro-localisation et Ferro-fiabilisation expliqués par la suite.

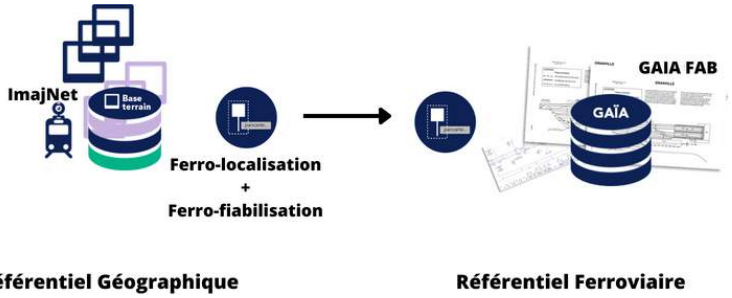


FIGURE 1 : Schéma récapitulatif de l'enjeu correspondant au changement de référentiel.

0.0.4 Objectif

L'objectif final de l'équipe I2D est donc de récupérer les images des ESV afin de croiser les données et d'obtenir des corrections si nécessaire pour avoir une base digitale la plus proche de la réalité terrain. Ainsi, l'objectif du stage est de trouver une méthode de détection des voies la plus effective possible pour répondre à ces enjeux.

0.0.5 Démarche

Afin de répondre à la problématique et ainsi atteindre l'objectif, deux approches sont étudiées : la première est l'utilisation du **Deep Learning** et la deuxième est l'exploitation de la **morphologie mathématique**. Dans les deux cas, un premier travail de recherche et d'état de l'art a été mis en place. Puis, différents essais ont été effectués sur des cas différents jusqu'à trouver toutes les hypothèses pour pouvoir trouver une solution la plus pérenne possible.

Chapitre 1 Fédération du patrimoine numérique et terrain

Les images permettent d'avoir une vision du terrain et c'est sur celles-ci que la détection des objets et des voies s'effectue. Ainsi, le processus de collecte des images est une étape primordiale et son traitement est important pour pouvoir les manipuler par la suite.

1.1 La collecte des données

La SNCF possède trois Engins de Surveillance de Voies ESV qui circulent sur 80% des lignes ferroviaires françaises. Ces trains s'occupent de la maintenance des voies. Dans le cadre de nos projets, les ESV filment leurs trajets. En effet, ils sont équipés d'une caméra à l'avant et à l'arrière, d'un GPS et d'une centrale inertielle. Les caméras permettent donc de filmer les trajets de chaque ESV. Le GPS permet d'avoir la position géographique des ESV. La centrale inertielle permet de garder un trajet le plus cohérent possible si la connexion du GPS est perdue, souvent lors des passages sous les tunnels. Le fournisseur Imajing récupère les vidéos des trajets des missions réalisées par les ESV avec toutes les informations nécessaires : coordonnées GPS en WGS84 et Lambert93, le roll, le pitch et le yaw de la caméra ou encore la date de la mission. Imajing envoie mensuellement un disque de 5 To qui contient toutes les missions effectuées dans le mois. Une vidéo contient environ 2500 images séparées tous les 3 mètres dont les points GPS de chaque image et les autres métadonnées de celles-ci sont récupérés. Un disque contient 2500 vidéos. Cela fait au total 6 250 000 images. Les images sont ensuite traitées à I2D. C'est ainsi que la nouvelle source de données est obtenue. Imajing a mis en place un site sur lequel il est possible de voir les images de toutes les missions avec les informations sur la date et la position du train sur le référentiel ferroviaire. Le site se nomme Imajnet et il y a deux vues : la cartographie des voies ferrées sur lesquelles les ESV sont déjà passées en France et la vision de bord à partir d'une des deux caméras du train (avec la possibilité de changer de vue de caméra) (figure 1.1). Il est possible de se déplacer sur les voies comme si le train se déplaçait ou alors sélectionner un endroit en particulier en saisissant une ligne, une voie et un point kilométrique ou le nom d'une station ou d'une ville par exemple.

Figure 1.1 : Il est possible de voir les missions selon différentes dates de vue proposées (Imajnet)

1.2 Intégration et formatage de la donnée

En plus du référentiel, la SNCF a sa propre cartographie : c'est la cartographie du réseau ferré, appelée aussi *cartographie des voies*. Elle est mise à jour régulièrement et permet de répertorier toutes les voies de l'infrastructure ferroviaire sur le plan géographique et ferroviaire. Une voie est composée de plusieurs Tronçons Itinéraires de Voie (TIV), c'est l'entité la plus petite de la cartographie des voies (figure 1.3a). Les TIV sont délimités par un PK début et un PK fin et ont un identifiant unique.

1.2.1 Ferro-localisation : une fédération des référentiels ferroviaire et géographique

Lors de la collecte des données, on récupère les positions géographiques des images. A l'aide d'un logiciel de SIG (QGIS, ArcGIS), il est possible de superposer les points GPS des images et la cartographie des voies (figure 1.2). Le principe de la *Ferro-localisation* consiste à affecter le point GPS au TIV le plus proche par projection orthogonale (figure 1.3b). Les TIV sont aussi appelés GeoTIV dans cette partie car ils sont géolocalisés.

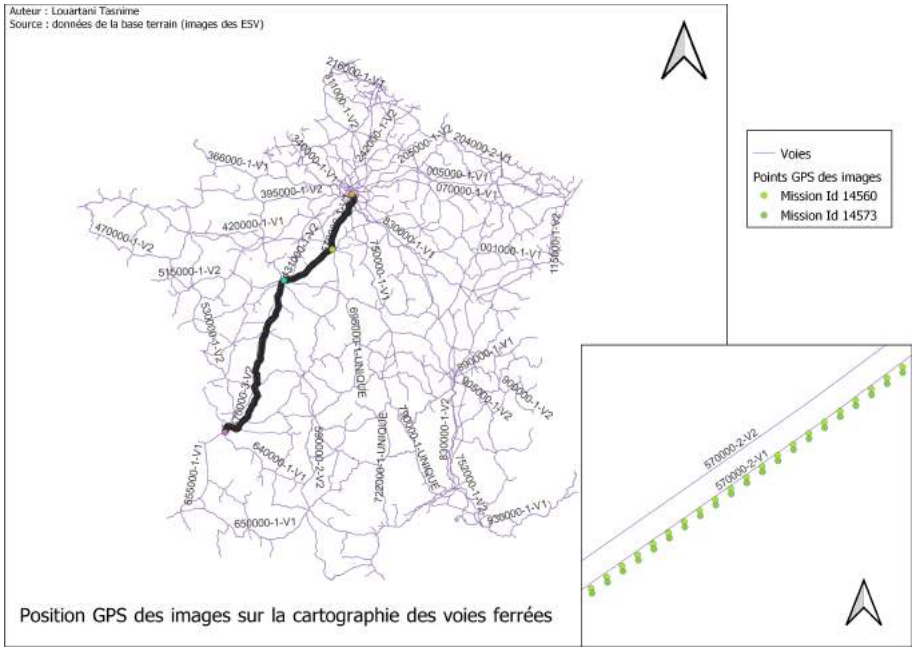
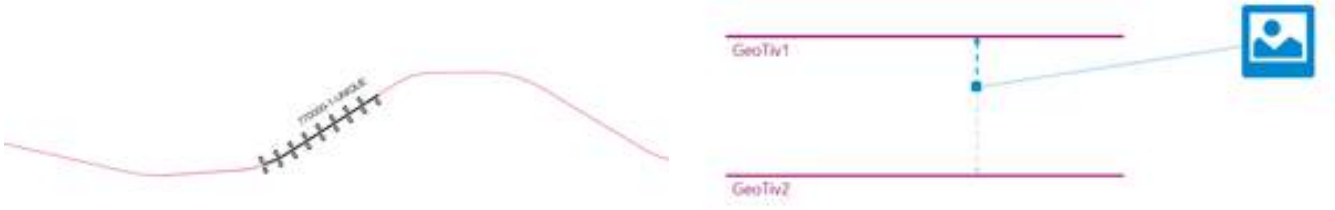


FIGURE 1.2 : Les points GPS des images sont affichés dans QGIS en les superposant avec la cartographie des voies.



(a) Un TIV de la voie unique de la ligne 770000

(b) Le point GPS (en bleu) est affilié au GeoTIV1 car il est plus proche que le GeoTIV2

FIGURE 1.3 : Document sur la Ferro-fiabilisation, I2D, 2023

1.2.2 Ferro-fiabilisation : la fiabilisation de la fédération entre deux référentiels

Etant donné que le GPS donne une position à 5 mètres près voire plus lors de passages dans les tunnels, les points peuvent être mal géolocalisés et donc affectés au mauvais TIV, ce qui créé un chemin incohérent. Ainsi la Ferro-localisation nécessite une correction : la *Ferro-fiabilisation*. En analysant le tracé des points GPS sur QGIS et d'après les métadonnées des images, il est possible de savoir lorsque le trajet de l'ESV est incohérent : s'il passe d'une voie à une autre en l'absence de voies de jonction par exemple.

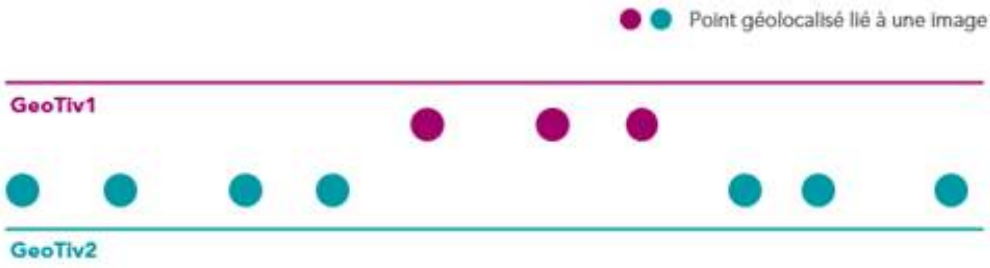


FIGURE 1.4 : Les points GPS ne forment pas un tracé cohérent à cause d'un changement de TIV aberrant (Document sur la Ferro-fiabilisation, I2D, 2023)

Les étapes de la Ferro-fiabilisation sont les suivantes :

- Pour chaque point GPS, il faut tout d'abord récupérer les 5 GeoTIV les plus proches dans un rayon de 15 mètres.
- Ensuite, il faut définir l'ensemble des chemins possibles. Il y a des règles métiers à appliquer pour pouvoir considérer qu'un chemin est cohérent comme par exemple un TIV ne peut être parcouru qu'une seule fois et dans le sens du parcours de l'ESV (figure 1.5).
- Parmi ces chemins, celui présentant l'offset le plus faible est sélectionné. L'offset correspond à la somme des distances entre les coordonnées GPS des images et les géolocalisations sur le GeoTIV (figure 1.6).
- Enfin, il faut rattacher les points aux GeoTIV retenus en fonction du parcours.

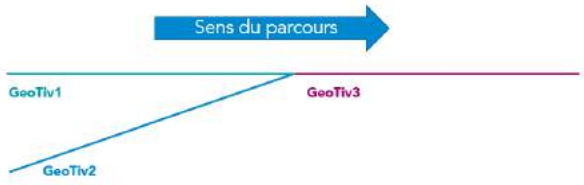


FIGURE 1.5 : Le chemin allant du GeoTIV1 au GeoTIV2 est impossible.

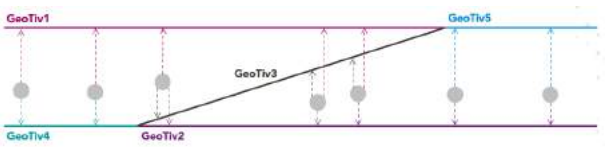


FIGURE 1.6 : Le chemin avec le moins d'offset est celui allant du GeoTIV4 au GeoTIV2.

Malgré ces étapes, il reste des passages d'une voie à une autre encore incohérents. Les cas les plus fréquents se trouvent lorsque l'ESV passe sous un tunnel. Le GPS ne fonctionne plus correctement sous un tunnel, d'où l'intérêt d'une centrale inertielle. Elle permet de garder une localisation

approximative lorsque le GPS ne reçoit plus de signal. Or, le dispositif peut être défaillant et provoquer un mauvais positionnement. Pour cela, des règles s'appliquent sur ces bouts de mission. Elles permettent de lier les résultats avant et après les tunnels pour déterminer la position dans le tunnel. Parfois, il est impossible de se baser sur les points en dehors du tunnel et donc la règle ne s'applique pas. L'affectation peut être sur une mauvaise voie sur toute la longueur du tunnel. Donc, il y a des cas problématiques qui nécessitent une vérification même après la Ferro-fiabilisation.

1.3 La détection automatisée d'objets dans les images

L'objectif de la détection automatisée des objets est de **corriger le du patrimoine digital**. En effet, I2D corrige les données saisies dans GAIA par la GAIA FAB. Pour cela, l'intelligence artificielle est utilisée comme étant une solution moins coûteuse et plus régulière sur les erreurs afin de détecter automatiquement les objets de l'infrastructure ferroviaire.

1.3.1 Fédérer le patrimoine numérique et terrain

La fédération du patrimoine numérique et physique constitue la feuille de route nommée *Loop'in*. Il s'agit de concilier les données du Système d'Information et la vision terrain de l'infrastructure. Tout d'abord, la GAIA FAB prend des documents sources (les plans techniques, les renseignements techniques...) et a un outillage qui permet d'administrer une base de données nationale nommée GAIA (étape 1 dans la figure 1.7). GAIA correspond donc à la description de l'infrastructure ferroviaire. Dans cette base, il existe des erreurs diverses : erreurs de saisie, documents sources qui ne sont pas à jour, problèmes d'outillage qui transforment la donnée... Comme I2D a besoin des données descriptives du réseau, l'équipe intervient aussi dans la correction de GAIA. Ainsi, son objectif est d'avoir une base numérique au plus proche de celle de la réalité terrain. Pour cela, il a été décidé de récupérer les images des ESV et détecter automatiquement les objets de l'infrastructure sur ces images (étape 2 dans la figure 1.7). Une fois que les objets sont détectés, ils doivent être localisés, c'est-à-dire affiliés à une voie. Le travail suivant consiste à croiser les données : celles obtenues par la détection automatique et celles saisies dans GAIA par la GAIA FAB(étape 3 dans la figure 1.7). En comparant les données, il est possible de voir des anomalies et en faisant un contrôle métier (étape 4 dans la figure 1.7), il est aussi possible de détecter l'origine des erreurs. Si l'erreur provient de la base GAIA, alors un travail de correction est appliqué (étape 5 dans la figure 1.7).

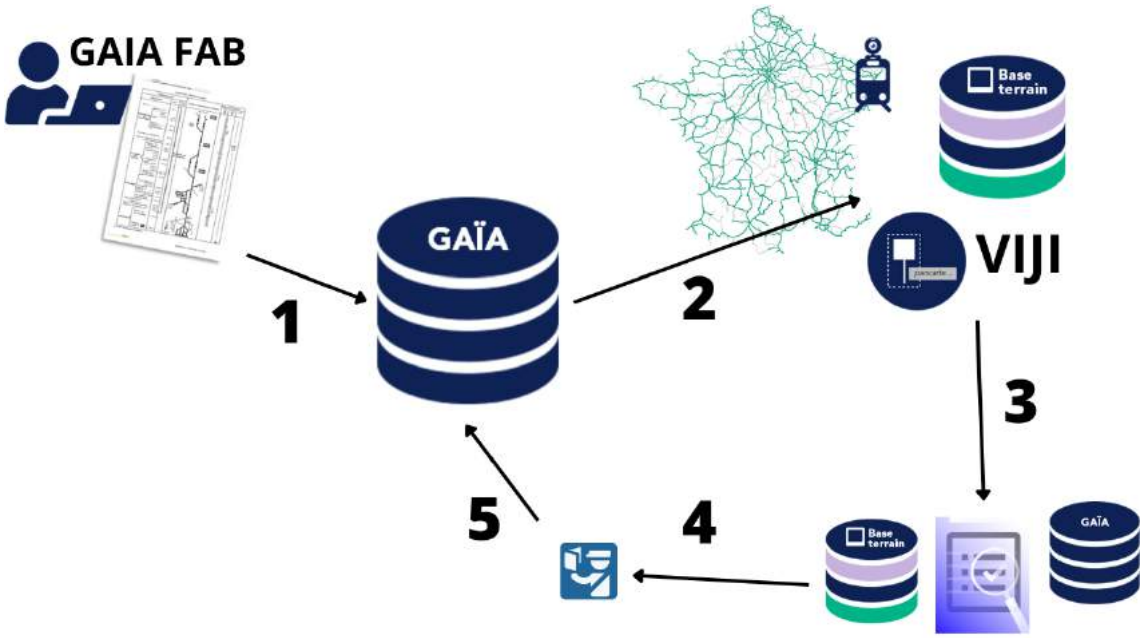


FIGURE 1.7 : Schéma de la Road Map Loop'in

1.3.2 La détection des voies : une réponse aux enjeux de numérisation

La détection automatique des objets s'appuie sur la technologie de l'intelligence artificielle : le Deep Learning. C'est une méthode qui malheureusement présente des limites. En effet, si une image contient plusieurs objets à localiser : comment connaître la voie qui est associée à cet objet ? C'est dans ces cas que le métier est important car, en connaissant les indications, il est possible de connaître les voies des objets. Par exemple, en comparant la distance entre les voies et l'objet, il est possible de savoir sur quelle voie l'objet est affilié. Cependant, il est question de créer une solution logicielle qui nécessite rarement d'intervention humaine. Pour cela, la détection automatique des voies permet de répondre à cet enjeu. La détection des voies permet de positionner correctement les objets aux voies. En détectant les voies sur l'image, il est possible de déterminer la distance entre les pancartes et les voies et ainsi savoir à quelle voie appartient un objet. A partir d'une voie, une abscisse curviligne peut être tracée : il s'agit de la courbe au milieu des deux rails d'une même voie, nommé fil de rail. En faisant une projection orthogonale du fil de rail aux objets, les distances entre les objets et la voie peuvent être calculées. Si la distance entre un objet et le fil de rail est faible, alors l'objet est affecté à cette voie.

Chapitre 2
Etat de l'art

Les images des ESV représentent le champ de vision du conducteur de train : voie sur laquelle le train circule ou parallèles ou de jonctions. Sur la même idée que la détection des objets, les voies sont des objets dans l'image. Ce sont des motifs qui se répètent dans une image. L'apprentissage automatique est donc une méthode intéressante à mettre en place et plus particulièrement le Deep Learning. De plus, les rails sont des lignes courbées. Dans ce cas, une autre approche est retenue : la détection de lignes dans les images. Il se trouve qu'un travail a déjà été mené sur cette approche : il s'agit de MorphoMat.

2.1 Deep Learning : des travaux autour de la détection des rails

Le Deep Learning est une technologie récente mais la problématique de détection des voies n'est sûrement pas méconnue dans le domaine de l'ingénierie des transports. En effet, des travaux de recherche existent depuis des décennies autour des enjeux de maintenance, de sécurité ou d'innovation tel que les trains autonomes par exemple. Dans cette partie, la diversité de ces travaux est mise en avant, notamment avec la présentation des différents modèles réalisés ces dernières années.

2.1.1 La segmentation des images à l'aide de l'outil SAM

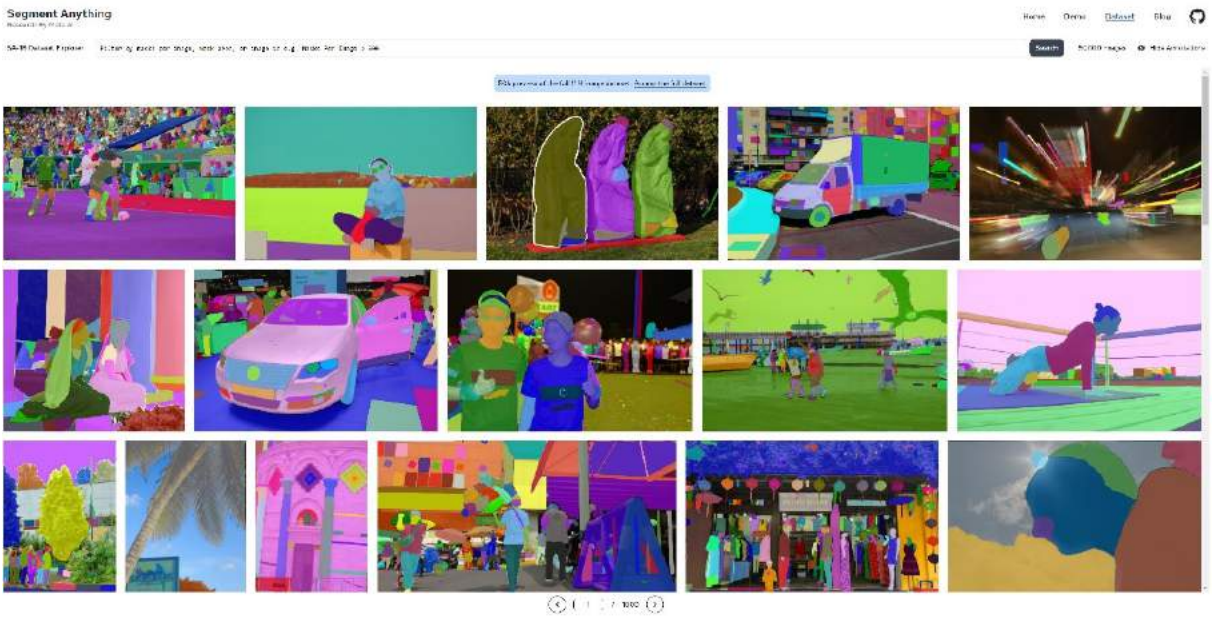


Figure 2.1 : Il est possible de voir le jeu de données utilisé par Meta AI pour créer l'outil SAM sur le site Segment Anything

La segmentation sémantique consiste à découper une scène en plusieurs parties correspondant à un objet ou un groupe d'objets similaires. C'est une méthode de traitement des images très exploitée

dans les travaux de recherche autour des enjeux de l'infrastructure ferroviaire (Feng et al., 2021). Il est dit que la segmentation des images est une avancée dans le domaine de la vision par ordinateur et que les algorithmes de segmentation d'images basés sur les réseaux de convolution du Deep Learning sont de plus en plus étudiés (Zhang et al., 2023). Ainsi, un des modèles retenus pour la suite est l'outil Segmentation Anything Model (SAM), réalisé par l'équipe de recherche Meta AI. L'idée est de travailler avec SAM car il présente des avantages de la segmentation d'instance. Tout d'abord, SAM fonctionne sur le principe de la segmentation rapide. Le jeu de données utilisé pour entraîner le modèle se nomme SA-1B (figure 2.1). Il comporte 11 millions d'images diverses et d'un 1 billion de masques annotés (Kirillov et al., 2023).

2.1.2 RailNET : un projet lié à l'infrastructure ferroviaire chinoise

Dans un but d'innovation, les trains autonomes ont besoin d'identifier de la manière la plus effective possible les obstacles sur les voies pour éviter tout risque d'accident. Pour cela, la détection des rails à partir d'images à vision de bord est une solution proposée. Il existe beaucoup de modèles tels que ResNet, ResUNet++ DFA-UNet ou encore DFF-UNet depuis le début de ces recherches (Ye et al., 2021, Ye et al., 2020, Zhang et al., 2023). Cependant, la détection des voies est complexe en raison de la grande diversité de l'infrastructure et des contraintes de prises des images. Mais surtout, il n'y a aucun jeu de données en libre-service correspondant à des images des infrastructures ferroviaires mis en place depuis l'arrivée du Deep Learning. Un modèle a été retenu par des chercheurs travaillant sur la détection des rails du réseau ferré en Chine : RailNet. C'est un algorithme de segmentation par Deep Learning pour la détection des voies ferrées qui est performant et plus rapide. Afin d'entraîner ce modèle, le jeu de données Railroad Segmentation Dataset (RSDS) a été créé car il n'en existait aucun auparavant (Wang et al., 2019). Il contient 3000 images dont 2500 pour l'entraînement, 200 pour la validation et 3000 pour les tests. Une équipe de chercheurs chinois a mis en place un autre jeu de données en 2022. Il contient 7432 images et prend en compte la diversité des images telles que les conditions météorologiques ou encore la complexité des rails : il s'agit de RailDB. Une fois que les images sont collectées dans RailDB, les rails sont labellisés à l'aide de polylignes. Ainsi, le modèle RailNet a été évalué sur ce jeu de données et les résultats sont assez démonstratifs : 96.22% de précision (Li et Peng, 2022).

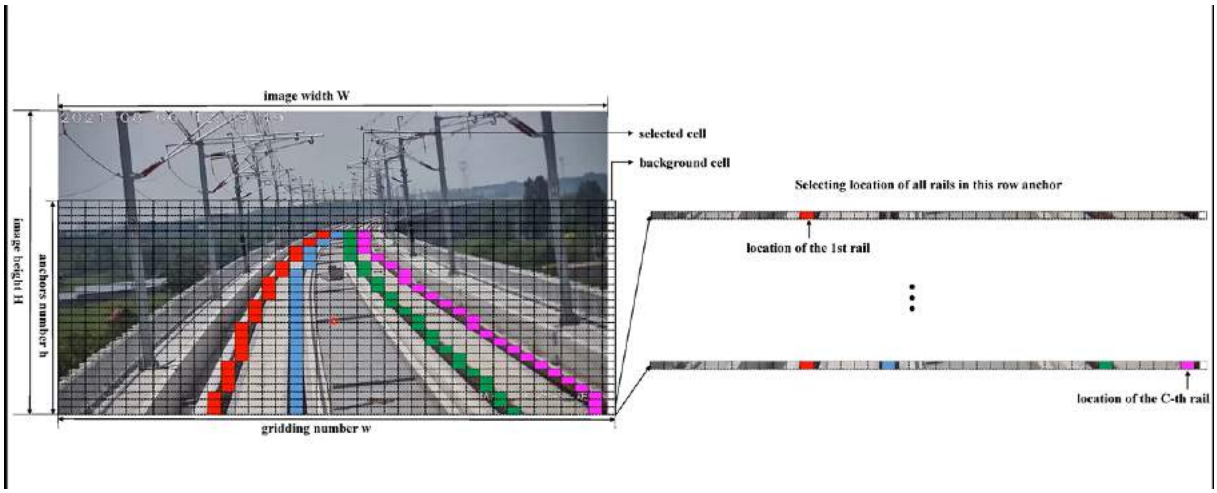


Figure 2.2 : L'image est séparée en plusieurs colonnes et lignes et chaque pixel est classifé. (Li et Peng, 2022)

2.2 Deep Learning : des modèles avec un jeu de données à définir

Le traitement des images en Deep Learning requiert l'utilisation d'algorithmes d'intelligence artificielle qu'on appelle Réseau de Neurones Convolutif (RCNN). Il y a différents modèles de RCNN dont deux qui sont retenus pour trouver une méthode de détection de voies : Mask RCNN et Faster RCNN. Ces deux modèles nécessitent la création d'un jeu de données à partir des images des ESV.

2.2.1 Utilisation du modèle Faster RCNN

La détection automatique d'objets se base sur la reconnaissance à l'aide d'une classification et la localisation des objets sur une image à l'aide d'encadrements appelés bounding boxes (Feng et al., 2021). Ce modèle a l'avantage d'être rapide à l'aide de l'optimisation des zones d'intérêt proposées pour détecter les objets dans une image. Un jeu de données doit être mis en place : il est composé d'images des ESV en prenant en compte différentes contraintes expliquées par la suite. Seule la labellisation est différente : elle se fait à l'aide d'encadrements nommés bounding box.

Figure 2.3 : Détection automatique des objets utilisant le modèle Faster RCNN développé par l'équipe I2D.

2.2.2 Utilisation du modèle Mask RCNN

Mask RCNN est un modèle qui permet de détecter des objets et de les classifier. En plus de la détection, il y a la segmentation d'instance, ce qui est avantageux par rapport aux autres modèles de détection car cela affine le résultat. La segmentation d'instance permet d'associer un masque et une annotation à chaque objet. Mask RCNN est donc une extension de Faster RCNN : il permet d'obtenir en plus les masques des objets (He et al., 2017). Pour l'entraîner, le jeu de données est le même mais la labellisation est différente : ce sont des polygones qui enveloppent les rails.

2.3 MorphoMat : un travail à optimiser

MorphoMat (morphologie mathématique) est le nom de la méthode commencée par un ancien consultant travaillant pour l'équipe de développement de la GAIA FAB. « [Elle] repose sur le principe consistant à comparer la structure inconnue, l'image que l'on étudie, à un ensemble de formes, ls éléments structurants dont on maîtrise toutes les caractéristiques, et cela, au moyen de relations booléennes telles l'intersection ou l'inclusion. » (Schmitt et Mattioli, 2013). En effet, il est question ici d'assembler les algorithmes de traitement des images permettant de détecter les rails. Dans les travaux de détection des voies, beaucoup citent l'utilisation d'algorithmes classiques de détection de contours dans les images, comme par exemple avec le filtre de Deriche (Dooze et al., 1998). Le traitement se fait sur une image à teinte de gris et floutée à l'aide d'une fonction gaussienne. Pour détecter les contours, un algorithme de Canny est appliqué et une région d'intérêt est définie afin de ne détecter que les rails situés au centre de l'image. Ces rails correspondent à la voie sur laquelle le train circule. Pour détecter les lignes dans l'image, un algorithme de Hough est appliqué (Kalms et al., 2017,Zhang et al., 2023). Enfin, après filtrage, les rails sont extraits et détectés dans l'image (figure 2.4). Cette méthode présente des irrégularités sur les résultats et connaît ses limites. C'est l'ébauche d'une étude sur laquelle le travail est simplifié : ce sont des images avec des cas simples et la zone d'intérêt est restreinte. Le but de la détection des voies est d'affilier les objets aux voies donc il est important de détecter toutes les voies sur toute l'image pour rattacher tous les objets de l'image aux bonnes voies.

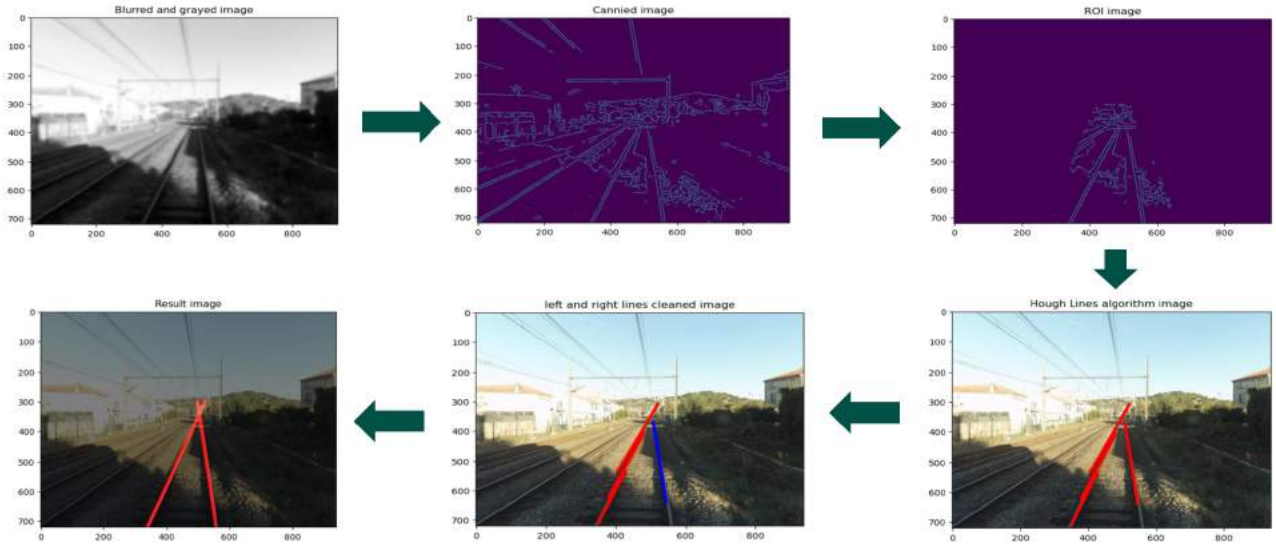


Figure 2.4 : Les étapes de MorphoMat sur une image d'un ESV

Chapitre 3
Applications

Afin de mettre en place les méthodes décrites dans l'état de l'art, il est nécessaire d'avoir un support informatique adapté. Le langage utilisé est Python version 3.8.10 et l'interface de développement est Visual Studio Code. Pour chaque méthode, différentes librairies ont été utilisées et sont présentées au cours de cette partie. Les librairies pour les calculs mathématiques de base ou pour la représentation graphique telles que numpy ou matplotlib.pyplot sont utilisés tout au long du stage. Pareillement, les librairies de traitement des images telles que cv2 se retrouvent tout le long de la détection des voies. De plus, les codes sont lancés dans un environnement virtuel. Les données utilisées pour lancer les codes des différentes approches sont les images des ESV.

3.1 Deep Learning : application des différents modèles et leurs résultats

Dans cette partie, les outils et algorithmes présentés dans l'Etat de l'art (chapitre 2) sont appliqués et travaillés en prenant compte des contraintes de support et de temps du stage.

3.1.1 Application de SAM pour la labellisation manuelle

Sur le site de Segment anything, il est possible de faire une démo avec une image proposée ou une image chargée depuis son ordinateur. La segmentation se fait aussi de manière interactive : les objets peuvent être extraits s'ils ont été segmentés par l'outil. Afin d'utiliser SAM à plus grande échelle sur les images des ESV, le code a été récupéré sur le github de l'équipe Meta AI avec le modèle pré entraîné. Le script est lancé sur 20 images quelconques récupérées sur le site Imajnet présentant des complexités différentes : multiplicité des voies, conditions météorologiques différentes, obstacles sur la caméra (goutte de pluie, essuie-glace, reflet du soleil), exposition à la lumière différente. L'objectif ici est de voir s'il est possible d'extraire les rails sur les images.

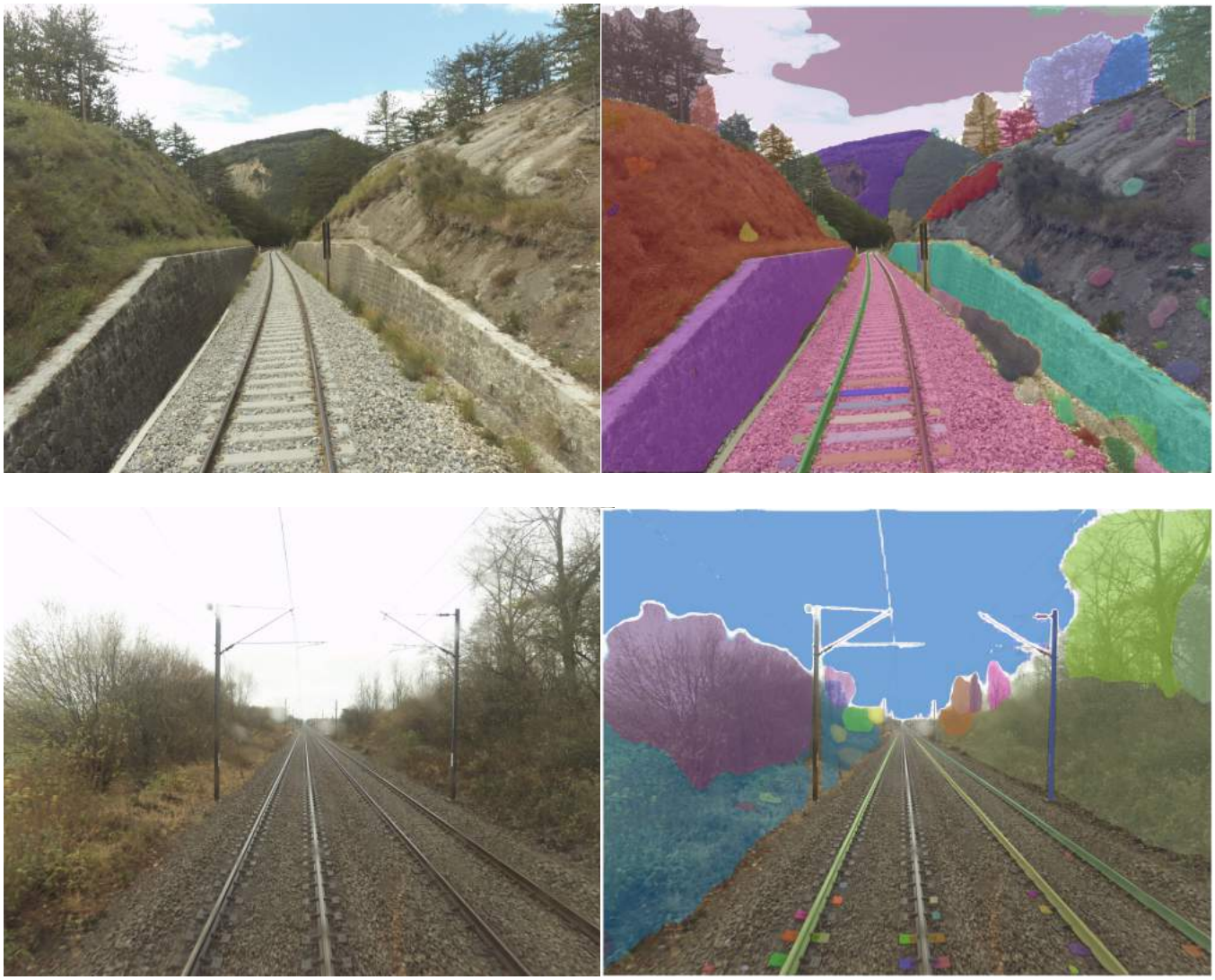


FIGURE 3.1 : Résultats de l'application de SAM sur des images d'ESV (à droite) : les masques sont représentés par couleurs.

Nombre de rails à segmenter	Nombre de rails segmentés par SAM
78	50

TABLE 3.1 : Tableau récapitulatif de l'application du modèle SAM sur 20 images d'ESV.

Dans les 20 images, il y a au total 78 rails à segmenter : les croisements entre rails sont négligés afin de simplifier la segmentation. Or, seulement 50 rails ont été segmentés soit environ 65% des rails ont été détectés par cet algorithme (Table 3.1). Cela reste insuffisant. De plus, les rails ne sont pas précisemment segmentés : soit ils sont inclus dans une voie, soit la segmentation est interrompue à cause d'un obstacle ou d'un croisement (figure 3.2). En analysant de plus près, la segmentation doit être beaucoup plus précise donc il est impératif de trouver une méthode d'amélioration. Pour se faire, il est possible de récupérer les masques sous la forme d'un fichier binaire sous format texte. Les masques correspondent au « 1 » sinon le reste de l'image correspond au « 0 » (figure 3.3). Afin de réfléchir à une correction, l'hypothèse pour la mettre en place est les suivante : les rails sont des lignes et partent du bas de l'image et vont jusqu'à un point de fuite qui se trouve plus ou moins au centre de l'image. Cependant, le temps de correction visant à améliorer la segmentation est trop important

pour se concentrer uniquement sur ce modèle. De plus, dans l'hypothèse de meilleurs résultats, il faut également trouver une méthode permettant à l'outil de distinguer parmi les objets qu'il a extrait lesquels correspondent des rails. Ainsi, l'outil SAM n'est plus privilégié car la labellisation manuelle est plus fiable.

FIGURE 3.2 : Certains rails ne sont pas segmentés après application de SAM (les endroits de l'image entourés en rouge montrent qu'il y a un obstacle qui gènent la segmentation en continu des rails).

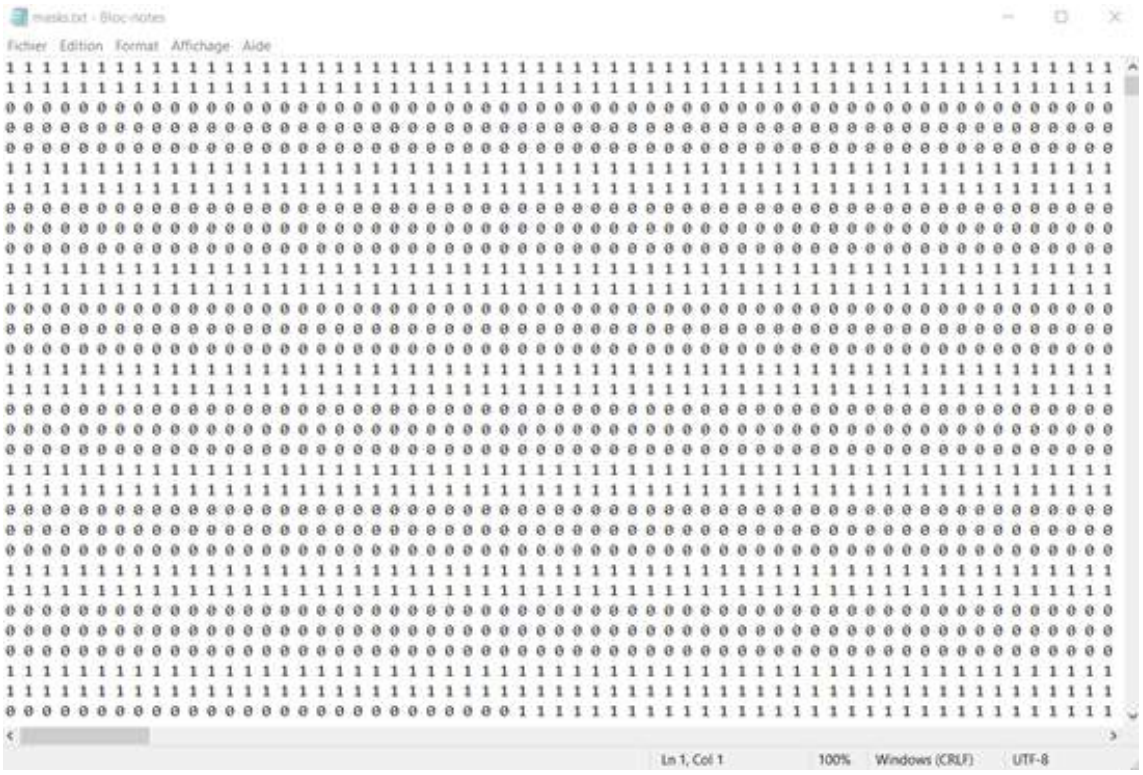


FIGURE 3.3 : Le résultat de l'application de SAM sur une image d'un ESV : un fichier texte représentant les endroits où il y a des masques à l'aide du binaire.

3.1.2 Application des modèles Mask et Faster RCNN

Afin d'appliquer les deux autres modèles de Deep Learning, il faut d'abord mettre en place un jeu de données d'images des ESV, appelé couramment dataset. Le dataset contient au total 60 images, toutes récupérées sur Imajnet (figure 3.4). Il est catégorisé selon le type de voies qu'il y a dans les images :

- Voie simple : il y a une seule voie dans les images
- Voies multiples : il y a 2 voies ou plus qui ne se croisent pas
- Voies complexes : ce sont des voies multiples mais avec la présence d'aiguilles ou autres types d'intersection

Pour chaque type de voie, il y a une précision sur la tendance linéaire du rail (colonne *«A l'horizon»* dans le tableau) : soit le rail parait droit, soit il fait une courbe parce qu'il y a un virage. De la même manière, pour chaque type de courbe, il y a des conditions météorologiques différentes :

- Beau temps : l'image est bien exposée à la lumière, aucun obstacle est présent devant la caméra (4 images pour chaque type de ligne) (figure A.1)
- Mauvais temps : en cas de pluie, brouillard ou neige car des gouttes d'eau se posent sur la caméra au niveau des rails (4 images pour chaque type de ligne) (figure A.2)
- Avec obstacles : présence d'un balais d'essuie-glaces (figure 3.5), d'une ombre, du reflet du logo de la caméra sur la vitre ou le reflet du soleil... (2 images pour chaque type de ligne)

Cela revient à 10 images pour chaque type de ligne. Comme il y a deux types de lignes par types de voies, il y a donc 20 images pour chaque type de voie, ainsi cela fait 60 images au total.

Types de voies	A l'horizon	Conditions	Nombre d'images pour le paramètre	Nombre d'images au total (pour un type de voie)
Voies Simples (2 rails pour une seule voie)	Ligne droite	Beau temps	4	20
		Mauvais temps	4	
		Avec Obstacles	2	
	Ligne courbée	Beau temps	4	
		Mauvais temps	4	
		Avec Obstacles	2	
Voies multiples (2 voies ou plus mais parallèles)	Ligne droite	Beau temps	4	20
		Mauvais temps	4	
		Avec Obstacles	2	
	Ligne courbée	Beau temps	4	
		Mauvais temps	4	
		Avec Obstacles	2	
Voies complexes (voies multiples et intersections ou autres configurations)	Ligne droite	Beau temps	4	20
		Mauvais temps	4	
		Avec Obstacles	2	
	Ligne courbée	Beau temps	4	
		Mauvais temps	4	
		Avec Obstacles	2	

FIGURE 3.4 : Tableau récapitulatif du jeu de données utilisé pour l'application des modèles Mask et Faster RCNN.

FIGURE 3.5 : Une partie des images présentes dans le jeu de données

Une fois que le dataset est construit (A), il faut annoter les rails sur ces images. Pour cela, l'outil VIA est utilisé. C'est un outil qui permet de faire la labellisation des images mais aussi des vidéos et d'audios. VIA signifie VGG Image Annotator avec VGG pour Visual Geometry Group. L'installation est simple : il suffit de télécharger un zip et, une fois installé, l'outil se lance sur une page html. L'outil permet de faire des annotations à l'aide d'encadrements, de cercles, d'ovales, de polygones, de points ou encore de polylignes (figure 3.6). Dans le cas du modèle Mask RCNN, les rails sont labellisés avec des polylignes dans un premier temps puis avec des polygones dans un deuxième temps.

FIGURE 3.6 : L'interface de l'outil VIA pour labelliser les rails.

Le modèle Mask RCNN est testé avec le modèle Faster RCNN car, d'après l'état de l'art, il est censé être plus performant. Le code est récupéré sur un github. L'entraînement est ensuite lancé et

le modèle est un fichier sous format *«.h5»* (figure 3.7) et est récupéré dans les logs. Il est ensuite lancé sur des images pour voir le résultat.

mask_rcnn_object_0001.h5	19/05/2023 11:57	Fichier H5	258 101 Ko
mask_rcnn_object_0002.h5	19/05/2023 12:03	Fichier H5	258 101 Ko
mask_rcnn_object_0003.h5	19/05/2023 12:10	Fichier H5	258 101 Ko
mask_rcnn_object_0004.h5	19/05/2023 12:16	Fichier H5	258 101 Ko
mask_rcnn_object_0005.h5	19/05/2023 12:22	Fichier H5	258 101 Ko
mask_rcnn_object_0006.h5	19/05/2023 12:28	Fichier H5	258 101 Ko
mask_rcnn_object_0007.h5	19/05/2023 12:34	Fichier H5	258 101 Ko
mask_rcnn_object_0008.h5	19/05/2023 12:41	Fichier H5	258 101 Ko
mask_rcnn_object_0009.h5	19/05/2023 12:48	Fichier H5	258 101 Ko
mask_rcnn_object_0010.h5	19/05/2023 12:54	Fichier H5	258 101 Ko
mask_rcnn_object_0011.h5	19/05/2023 14:28	Fichier H5	258 101 Ko
mask_rcnn_object_0012.h5	19/05/2023 14:34	Fichier H5	258 101 Ko
mask_rcnn_object_0013.h5	19/05/2023 14:40	Fichier H5	258 101 Ko
mask_rcnn_object_0014.h5	19/05/2023 14:48	Fichier H5	258 101 Ko
mask_rcnn_object_0015.h5	19/05/2023 14:55	Fichier H5	258 101 Ko
mask_rcnn_object_0016.h5	19/05/2023 15:02	Fichier H5	258 101 Ko
mask_rcnn_object_0017.h5	19/05/2023 15:10	Fichier H5	258 101 Ko
mask_rcnn_object_0018.h5	19/05/2023 15:17	Fichier H5	258 101 Ko
mask_rcnn_object_0019.h5	19/05/2023 15:25	Fichier H5	258 101 Ko

FIGURE 3.7 : Les fichiers de logs sont enregistrés à chaque itération lors de l'entraînement du modèle.

FIGURE 3.8 : Après application du modèle Mask RCNN, les masques (en couleur) ne correspondent pas aux rails.

Comme les résultats avec les polylignes ne donnent pas des résultats concluants (figure 3.8), les étapes précédentes sont réitérées mais avec le jeu de données labellisé à l'aide de polygones.
Les résultats ne sont toujours pas concluants. Lors du lancement du modèle, une erreur apparaît à chaque fois (figure 3.9) donc l'hypothèse est de dire que c'est la raison pour laquelle les résultats ressemblent aux images obtenues comme sur la figure 3.8.

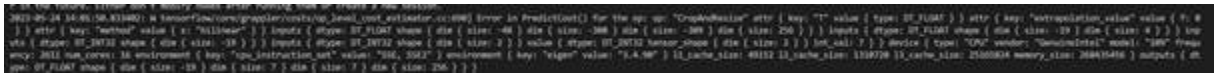


FIGURE 3.9 : L'erreur se trouve dans la prédiction et se nomme « Crop and Resize »

Après avoir cherché la provenance de cette erreur, il s'agit probablement d'un problème de redimensionnement. Par manque de temps et de solution, ce modèle est mis de côté.
Comme la détection automatique des objets mise en place par I2D repose sur le modèle Faster RCNN, l'idée est ici de reprendre les travaux. Tout d'abord, le jeu de données doit être labellisé de manière différente. En effet, les rails doivent être représentés par des bounding boxes. Un autre outil est déjà utilisé par l'équipe pour la labellisation, c'est LabelIMG. Il ne permet uniquement que de faire des encadrements rectangulaires contrairement à VIA (figure 3.10). A partir de cette constatation, la question déterminante est la suivante : comment un rail, à savoir une droite, peut être labellisée à l'aide de cadres ? Deux méthodes sont appliquées : la première consiste à représenter un rail à l'aide d'un seul rectangle et la deuxième consiste à labelliser un rail à partir de plusieurs bounding boxes. L'entraînement est lancé avec la première méthode et les résultats sont décevants car les rails ne sont pas correctement détectés (figure 3.11a). En effet, le modèle constate une précision très faible (environ 0.1).

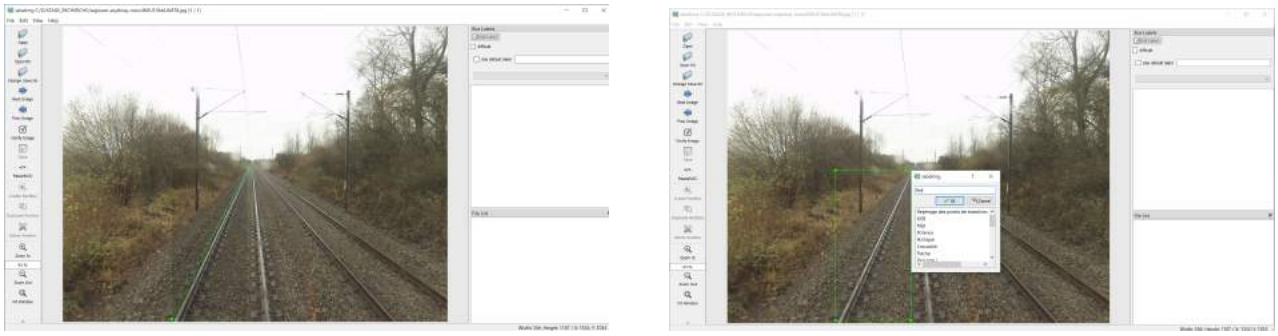


FIGURE 3.10 : L'interface de l'outil LabelIMG permettant de faire de labelliser les rails à l'aide de bouding boxes.

Le fait qu'une seule bounding box correspond à un rail implique beaucoup de bruit. Ainsi l'entraînement est lancé mais cette fois avec la deuxième méthode de labellisation. Les rails ne sont pas non plus correctement détectés et la précision est très faible comme vu sur l'image (figure 3.11b). Il y a plusieurs hypothèses à cet échec : bruit trop élevé, entraînement effectué sur un nombre insuffisant d'images, problème de dimensionnement d'images identique à celui appliqué sur le modèle Mask RCNN. Par manque de temps et de solution, **ces modèles sont mis de côté.**

(a) Un rail est labellisé avec une seule bounding box.

(b) Un rail est labellisé avec plusieurs bounding boxes.

FIGURE 3.11 : Résultat de l'application du modèle Faster RCNN avec deux méthodes de labellisation différentes.

3.1.3 Application du modèle RailNet

Dans cette partie, il est question d'utiliser un modèle qui est adapté à nos types de données : des images de rails prises par des caméras embarquées à l'avant ou l'arrière d'un train en mouvement. Les codes sont récupérables sur le github de l'équipe de chercheurs qui a travaillé sur le modèle RailNet. Afin d'avoir accès au jeu de données RailDB, il faut remplir un formulaire lui indiquant l'intérêt de travailler sur ses données. Comme c'est un modèle pré entraîné, le jeu de données créé précédemment ne sert pas dans cette partie. Quelques unes des images de RailDB sont récupérées et le modèle est lancé sur celles-ci afin de vérifier si déjà avec ces images, le modèle donne des résultats corrects (figure 3.12). Les résultats montrent le bon fonctionnement du modèle (figure B.1).

FIGURE 3.12 : Application du modèle RailNet sur une image de RailDB (les rails sont en couleur).

Les images des ESV sont récupérées sur Imajnet et le modèle est appliqué. Théoriquement, les rails doivent être détectés comme c'est le cas avec les images de RailDB. Cependant, les résultats montrent que le modèle ne fonctionne pas du tout : les rails ne sont pas correctement quantifiés et n'ont pas la bonne forme (figure 3.13).

Pour confirmer que le problème ne provient pas des images des ESV, le modèle est appliqué sur des images prises sur Google Image. Les rails ne sont malheureusement pas détectés. Dès lors, le code de RailNet est inspecté pour comprendre l'origine de cette erreur allongeant ainsi le temps

FIGURE 3.13 : Application du modèle RailNet sur une image d'Imajnet (les rails sont en couleur.

consacré à ce modèle sur le stage. L'hypothèse de l'échec est la différence d'entrée : les images. Certes, l'entraînement est fait sur les images de RailDB, mais qu'est-ce qui différencie autant les images de RailDB de celles d'Imajnet ?

En suivant la réalisation du modèle à l'aide de l'inspection des scripts, il est question de normaliser les images. En effet, la normalisation est une étape importante lors du Deep Learning. Cela consiste à étendre la plage de valeurs de l'histogramme d'une image. Pour se faire, une transformation est appliquée en utilisant les valeurs par défaut à la moyenne et l'écart type des pixels des images du jeu de données ImageNet si les images utilisées pour l'entraînement sont similaires à ImageNet. C'est la fonction *torchvision.transforms.Normalize()* qui permet de faire la normalisation. Ce qui m'amène à calculer les valeurs de normalisation des images des ESV pour voir s'il y a une différence flagrante (figure 3.14 et figure 3.15). Peut être que si les valeurs de normalisation sont plus adaptées aux images des ESV en entrée, cela modifie le résultat du modèle. Sur la figure 3.14, les moyennes et écart types sont calculés sur les images des ESV pour chaque canal de pixels : ce qui donne une normalisation de $(0.532, 0.546, 0.520), (0.277, 0.293, 0.310)$. En parallèle, les valeurs de normalisation des images de RailDB sont aussi calculées : $(0.486, 0.499, 0.471), (0.188, 0.181, 0.205)$ (figure 3.15).

Valeurs de normalisation		
Images	Moyenne	Ecart-type
ImajNet	$(0.532, 0.546, 0.520)$	$(0.277, 0.293, 0.310)$
RailDB	$(0.486, 0.499, 0.471)$	$(0.188, 0.181, 0.205)$
ImageNet	$(0.485, 0.456, 0.406)$	$(0.229, 0.224, 0225)$

TABLE 3.2 : Tableau récapitulatif des valeurs de normalisation pour chaque image.

Ainsi, l'écart de valeurs de normalisation entre les images de RailDB et celles d'ImajNet est de $(0.046, 0.047, 0.049), (0.089, 0.112, 0.105)$. L'écart est minime mais n'est pas négligeable. A la suite de ces calculs, il est possible de confirmer que la normalisation joue un rôle dans l'échec de ce modèle sur les images d'ImajNet. Cependant, en analysant les valeurs de normalisation des images de RailDB et celles par défaut, utilisées dans le code : il y a aussi un écart. En effet, les valeurs de normalisation des images de RailDB sont $(0.486, 0.499, 0.471), (0.188, 0.181, 0.205)$ et celles par défaut sont $(0.485, 0.456, 0.406), (0.229, 0.224, 0225)$ (Table 3.2).

Il y a donc un écart de $(0.001, 0.043, 0.065), (0.041, 0.043, 0.020)$. De même, l'écart est minime mais n'est pas négligeable. Dans les deux cas, il y a un certain écart avec les données de normalisation

	IMAJNET	MEAN			STD		
		R	G	B	R	G	B
même image, la résolution est différente	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet2.jpg	0,6051	0,5781	0,5283	0,2837	0,3051	0,3348
	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet_96ppp.jpg	0,6034	0,5731	0,5643	0,2891	0,309	0,3235
	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet.jpg	0,6034	0,5731	0,5643	0,2891	0,309	0,3235
	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet3.jpg	0,5938	0,572	0,5445	0,3005	0,3141	0,3271
	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet4.jpg	0,5292	0,5217	0,4648	0,2586	0,2555	0,2589
image condition météo différente (plus ensoleillée)	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet5.jpg	0,3473	0,3689	0,3285	0,343	0,3361	0,3133
	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet6.jpg	0,5689	0,5536	0,5029	0,2721	0,2725	0,2933
	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet7.jpg	0,5253	0,5799	0,5544	0,2366	0,2712	0,3212
	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet8.jpg	0,4558	0,571	0,6396	0,248	0,2773	0,2752
	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet9.jpg	0,5146	0,5664	0,5262	0,2516	0,2796	0,3287
	C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/imajnet10.jpg	0,5807	0,5733	0,5471	0,2888	0,3069	0,3272
	moyenne	0,53241	0,5458	0,52006	0,2772	0,29273	0,31032
	médiane	0,54905	0,5715	0,5364	0,2779	0,29235	0,32235

FIGURE 3.14 : Tableau représentant les valeurs de normalisation pour chaque image récupérée d'ImajNet.

par défaut et pourtant cela fonctionne correctement voire parfaitement sur les images de RailDB. En conclusion, la normalisation n'est pas une piste concrète. Cependant, si les valeurs sont modifiées et que le modèle est ensuite appliqué sur les images de RailDB, les rails ne sont pas correctement détectés. De plus, en observant la distribution des pixels (l'histogramme) des images des ESV et celles de RailDB après normalisation avec les valeurs par défaut sur la figure B.2), les motifs sont différents : la fréquence des pixels est très grande entre 0.8 et 1.0 (figure 3.18a, figure B.3)) pour les images des ESV alors qu'elle est très faible pour les images de RailDB (figure 3.18b, figure B.2).

	MEAN			STD		
RAILDB	R	G	B	R	G	B
C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/example.jpg	0,5224	0,531	0,5516	0,1834	0,1777	0,1782
C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/fixed_rain_2_000005.jpeg	0,5165	0,5137	0,5048	0,1864	0,188	0,2019
C:/D/STAGE_RECHERCHE/Rail-Detection-main/utils/tele_normal_000006.jpeg	0,4179	0,4534	0,3578	0,1935	0,1778	0,2359
moyenne	0,4856	0,49936667	0,4714	0,18776667	0,18116667	0,20533333
médiane	0,5165	0,5137	0,5048	0,1864	0,1778	0,2019
IMAGENET VALEURS	0,485	0,456	0,406	0,229	0,224	0,225

FIGURE 3.15 : Tableau représentant les valeurs de normalisation pour chaque image récupérée de RailDB.

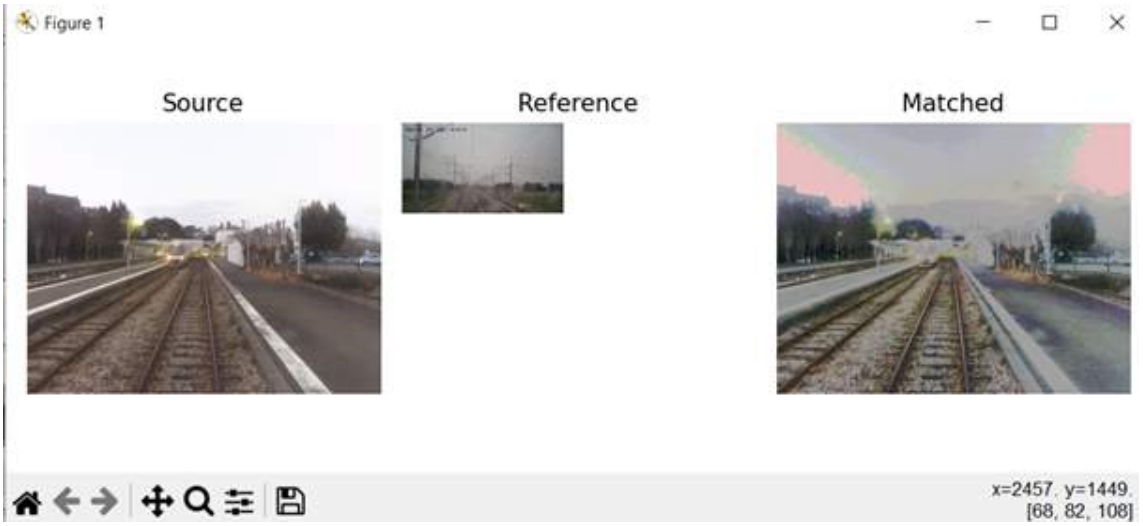
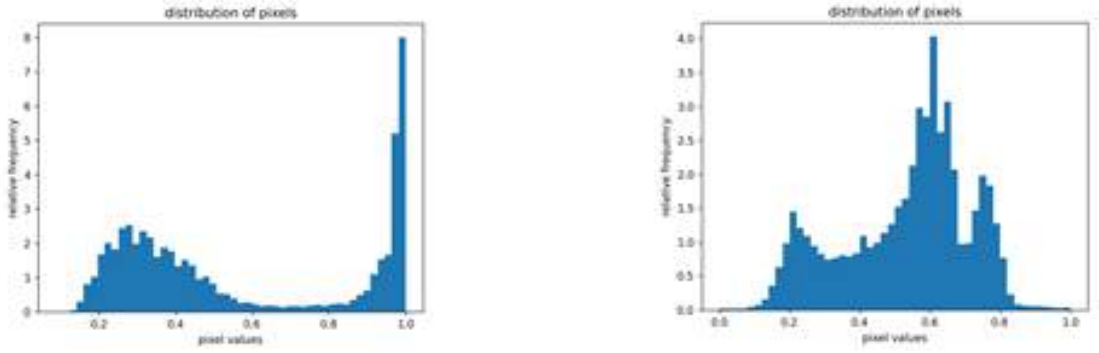


FIGURE 3.16 : Récapitulatif de l'étape «histogram matching».

FIGURE 3.17 : Résultat du modèle RailNet sur une image modifiée par l'«histogram matching».

Par manque de temps, une autre piste est travaillée en mettant celle-ci de côté. L'idée est de modifier les images prises en entrée du modèle à partir des distributions de pixels. Pour cela, il existe la méthode de *« matching »* des histogrammes. En effet, le modèle fonctionne sur les images de RailDB ayant une distribution de pixels particulière.



(a) Image d'ImajNet

(b) Image de RailDB.

FIGURE 3.18 : La distribution des pixels sur une image d'Imajnet et de RailDB.

Si les images ont des histogrammes semblables, peut-être que les rails sont mieux détectés. Le matching consiste à faire correspondre l'histogramme d'une image (la source) à partir de l'histogramme d'une autre (la référence) (figure 3.16). Ainsi, l'image source est modifiée. La librairie *skimage* contient la foncion *match_histograms*. Dans ce cas, les images sources sont celles d'ImajNet et les images de référence sont celles de RailDB. Sur la figure 3.19, la fonction est bien appliquée car on voit qu'il y a bien une présence du pic de pixels de chaque canal similaire à l'histogramme de référence sur les histogrammes de la colonne «Matched». De plus, l'image est modifiée comme on peut le voir sur la figure 3.17.

Il n'y a aucune différence entre les résultats d'avant et les résultats après modification : les rails ne sont pas correctement détectés (figure 3.17). Le temps de compréhension et d'optimisation du modèle est beaucoup trop important sachant qu'il est question de travailler sur la deuxième approche

qui n'implique pas de Deep Learning. Nous avons suggéré **l'hypothèse de surapprentissage** car en modifiant en entrée les images de RailDB, les rails ne sont pas correctement détectés.

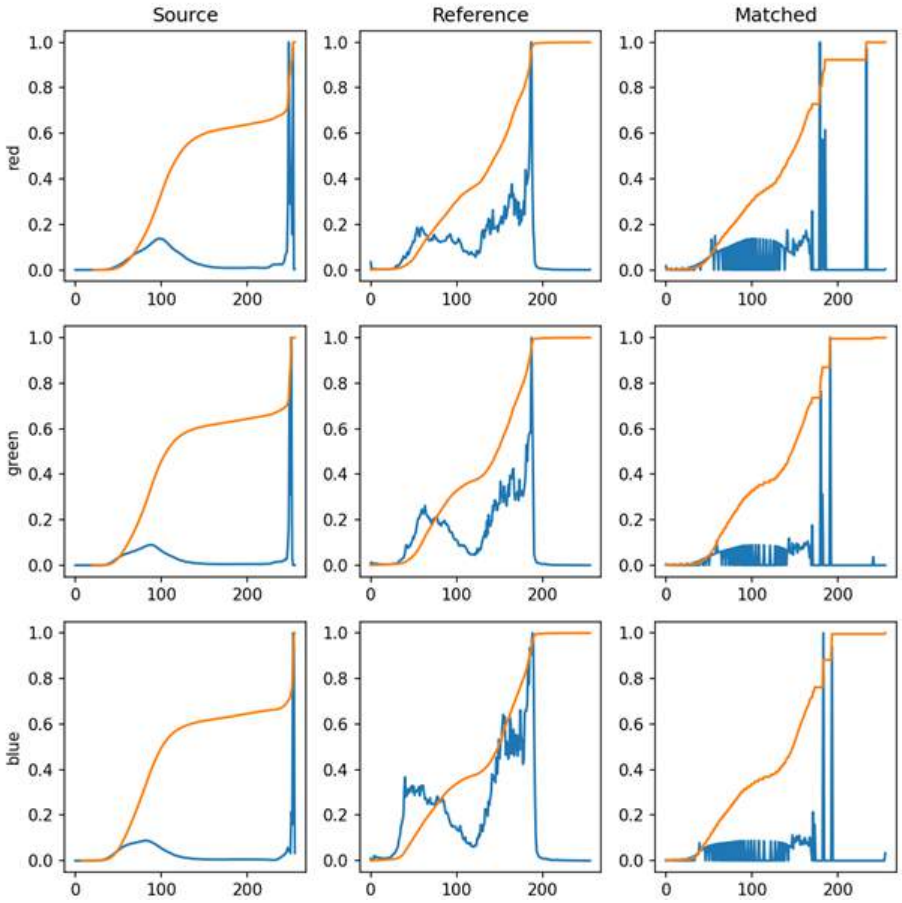


FIGURE 3.19 : Les histogrammes de chaque canal de pixel (rouge, vert, bleu) d'une image source, d'une image de référence et le résultat après «histogram matching».

3.2 Morphomat : optimisation de l'algorithme

A partir des travaux effectués sur MorphoMat, l'objectif est de voir le résultat sur des images des ESV afin de déterminer les axes d'améliorations. Il en résulte que :

- L'algorithme ne détecte pas tout le temps les rails même s'il détecte d'autres lignes sur l'image
- Il peut détecter plusieurs lignes autour du même rail, dû à l'ombre que le rail fait sur le sol
- Il peut aussi détecter des lignes au niveau du ciel
- Des lignes horizontales et verticales qui correspondent à d'autres objets que les rails (les poteaux, les murs...) sont aussi détectées

Par conséquent, il est nécessaire de filtrer les lignes détectées : il ne faut ni qu'elles soient horizontales ni verticales. Pour cela, une fois que les lignes sont détectées dans l'image, les paramètres a et b de chacune des droites sont récupérés ($y = ax + b$). Les pentes sont ensuite calculées à l'aide de la formule suivante :

$$\frac{y_1 - y_2}{x_1 - x_2}$$

```
# --- For every line detected we calculate the slope
        slope_dict = splot_lines (lines)

        slope_dict = { key : value  for key, value in slope_dict.items() if (value < -0.5
        and value > -15 and value != 0 ) or (value < 2.5 and value>0.25 and value!=0)  }

        lines =  [ line for   line in slope_dict.keys() ]
```

FIGURE 3.20 : Les valeurs de pente sont filtrées pour ne garder que les lignes pouvant correspondre à des rails. (extrait du code)

Le plus important est que les pentes ne doivent pas être égales à 0 ou aient une valeur très grande (figure 3.20). Ensuite, les lignes extraites après ce premier filtrage sont classifiées. Avec l'ombre et le fait que les rails sont gris clair au centre et gris foncé sur les bords, plusieurs lignes sont détectées au niveau des changement de couleur pour le même rail. Le but est de rassembler ces lignes pour le même rail. La classification se fait au niveau des valeurs de pente : si plusieurs lignes ont des pentes égales voire très proches, c'est sûrement le même rail. Or, des lignes peuvent être parallèles et ne pas correspondre au même rail. Donc, il faut aussi classifier selon les distances entre les lignes. Ainsi, si deux lignes ou plus sont pentues de la même manière et sont très proches alors c'est la même classe (figure 3.21).

FIGURE 3.21 : Les lignes sont classifiées selon leurs valeurs de pente et leurs distances entre elles.

Par ailleurs, pour que la méthode de filtrage des lignes fonctionne, il est impératif de ne détecter des rails qu'au niveau des voies. En effet, il y a parfois des cas où les caténaires sont détectés. En conclusion, la région d'intérêt (ROI) n'encadre plus toute l'image : elle est restreinte au niveau de la voie sur laquelle le train circule, la voie principale. Elle est d'ailleurs centrée au milieu de l'image (figure 3.22).

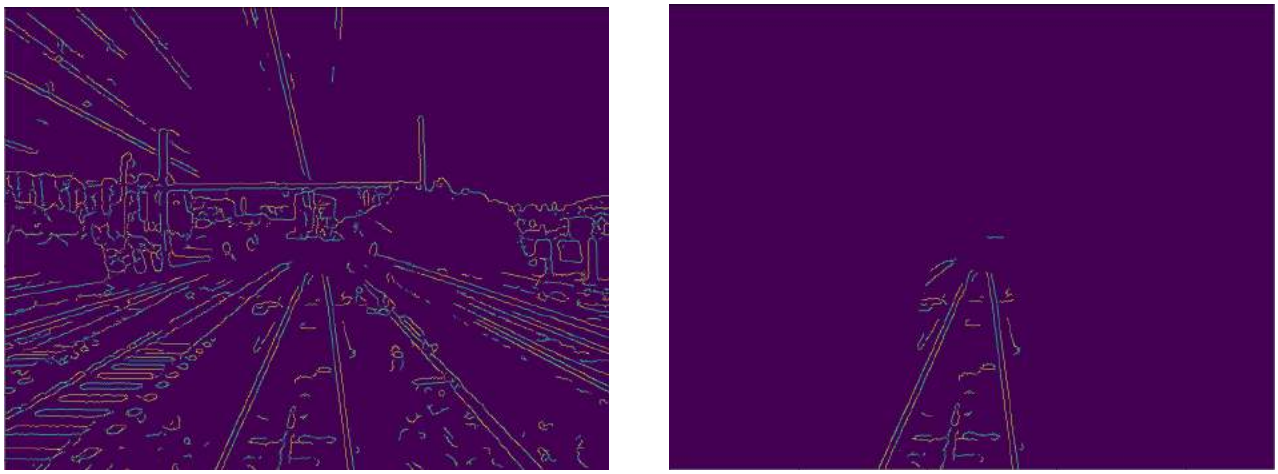


FIGURE 3.22 : La région d'intérêt est restreinte afin de ne détecter que les voies au milieu de l'image : la ROI se fait sur les images sur lesquelles l'algorithme de Canny est appliqué.

(a) Dans un premier temps, les rails sont détectés à l'aide des algorithmes.

(b) Ensuite, des demi-droites sont tracées jusqu'au bord de l'image pour qu'elles se croisent à un certain point.

FIGURE 3.23 : Les rails de la voie principale sont détectés (en rouge sur figure 3.23a puis tracés en continu figure 3.23b.

Une fois les filtres appliqués et la région d'intérêt fixée, il est possible de détecter les rails de la voie principale (figure 3.23). Cependant, il reste des cas où un rail sur deux voire aucun rail n'est détecté. Cela vient très probablement de l'exposition à la lumière qui fausse le changement de couleur et donc trompe la détection de contours et de lignes sur les images. Si les seuils des algorithmes de Canny sont modifiés, il n'y a pas de résultats réguliers sur toutes les images. En effet, si à chaque fois un problème a une solution qui s'adapte à seulement certaines images, l'algorithme n'est pas assez performant pour pouvoir l'appliquer à toutes les images des ESV. Il faut donc réfléchir à une autre méthode. C'est aussi une des raisons pour laquelle la détection des voies se concentre d'abord sur les rails de la voie principale. Lors d'une mission, les motifs se répètent : le train circule sur une voie qui se positionne à peu près au milieu de l'image et les rails de cette voie convergent vers un point de fuite également situé au même endroit sur l'image. Une mission correspond à une vidéo, soit 2500 images. Si sur 2500 images, il y au moins une image sur laquelle les deux rails de la voie principale sont détectés, alors ils sont considérés toujours au même endroit sur le reste des images de la même vidéo. Afin de mettre en place cette théorie, il est nécessaire de travailler sur plusieurs

images consécutives de la même vidéo. Pour débuter, 50 images sont récupérées de la base terrain. Sur chacune des images, une fonction est appliquée et permet de récupérer uniquement les rails de la voie principale. La fonction permettant de détecter les rails de la voie principale reprend les étapes expliquées précédemment dans l'état de l'art et dans cette partie :

- Lecture de l'image et application d'un filtre gaussien pour avoir une image en teine de gris et floutée
- Détection des contours grâce à l'algorithme de Canny et région d'intérêt fixée au niveau de la voie principale
- Détection des lignes avec l'algorithme de Hough
- Si au moins une ligne est détectée : application des filtres et extraction de ces lignes
- Récupération des paramètres de droites des lignes extraites

	A	B	C	D	E
1	Séquence	med_a_left	med_b_left	med_a_right	med_b_right
2	Vidéo 156	9,333333333	-4161,33333	-2,14285714	1395

FIGURE 3.24 : Le résultat de la médiane de tous les paramètres a et b des droites correspondant aux rails de la voie principale de chaque image d'une vidéo.

En conséquence, pour chaque image, les paramètres de droite a et b sont récupérés. Pour une vidéo, il suffit de calculer la médiane des paramètres de chaque image pour considérer que sur les 50 images consécutives, les rails de la voie principale sont toujours détectés (figure 3.24). Enfin, une fonction est développée pour prendre en entrée une vidéo et avoir en sortie les paramètres de droite moyens (**??**).

La deuxième partie de l'optimisation consiste à détecter les rails des voies restantes. Pour cela, nul besoin d'appliquer les algorithmes de détection des lignes et de fixer une zone d'intérêt. En observant les images et par principe de perspective, les voies parallèles à la voie principale se rejoignent au même endroit : le point de fuite défini précédemment (figure 3.25). Lorsqu'un cercle ayant pour centre le point de fuite est dessiné, alors les rayons peuvent coïncider avec les rails.

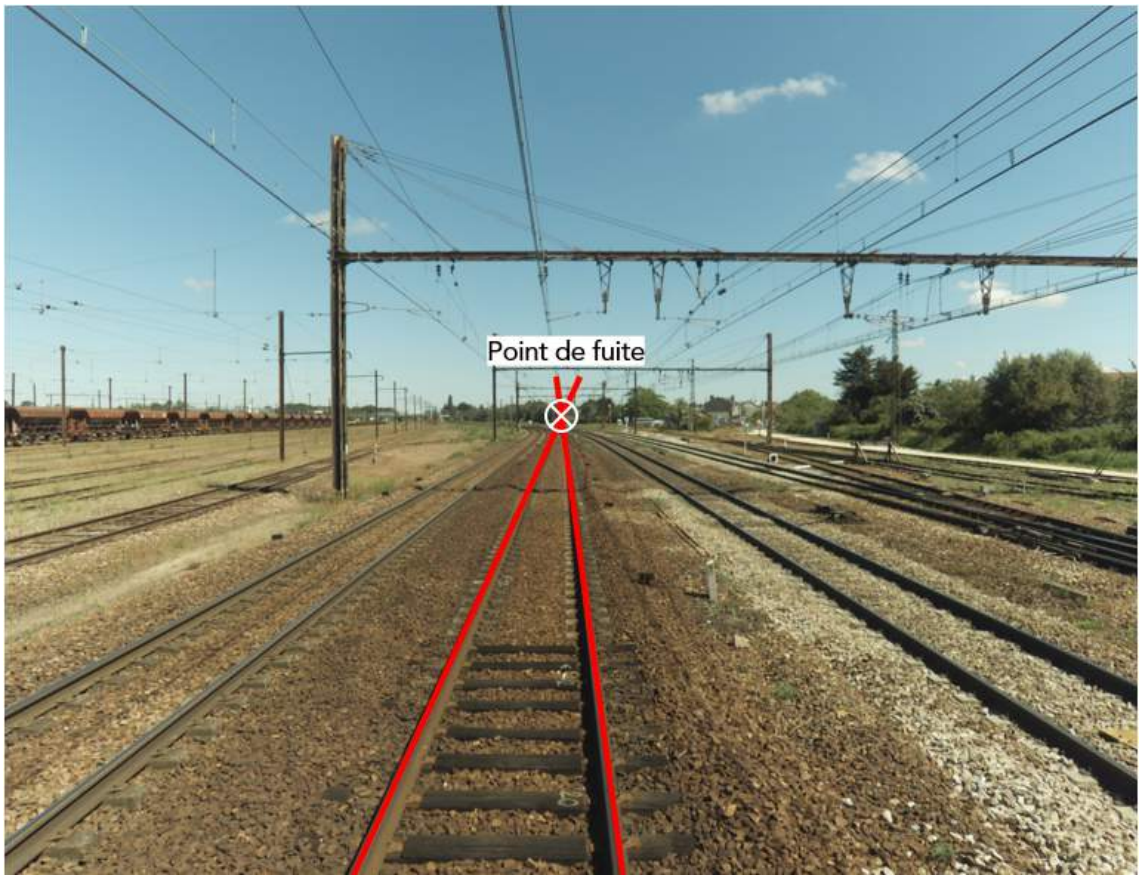


FIGURE 3.25 : Le point de fuite est le croisement des deux rails de la voie principale. Il est presque situé au milieu de l'image.

Un rail est une barre de fer de couleur grise quasi uniforme. Cela signifie que la différence entre les valeurs de deux pixels du rail est très faible. L'objectif est donc de projeter des lignes à partir du point du fuite jusqu'en bas de l'image pour voir si elles correspondent à des rails. Pour savoir si une ligne projetée correspond à un rail, il suffit de calculer la différence de valeurs entre les pixels au niveau de la ligne. C'est l'étape de l'analyse des deltas colorimétriques.
Pour mettre en place cette étape, il faut trouver la méthode de projection. La première idée est de fixer des angles entre les droites projetées et les rails de la voie principale. En effet, d'après les normes ferroviaires, l'écart entre les deux rails d'une voie est de 1435 mm et la distance entre deux voies parallèles ne change jamais sur les mêmes types de lignes ferroviaires. Par principe de perspective, les angles des rails se formant au niveau du point de fuite semblent être de même valeur. Entre les rails de la voie principale, il y a en moyenne 32 degrés. Entre le rail de la première voie parallèle à gauche de l'image et la voie principale, l'angle va de 25 à 34 degrés. Et entre le rail de la première voie parallèle à droite de l'image et la voie principale, l'angle va de 35 à 40 degrés (figure 3.26). Les angles varient de quelques degrés entre chaque image donc ce n'est pas assez précis pour retenir cette idée. Par contre, il est intéressant de retenir les valeurs pour avancer dans notre réflexion.

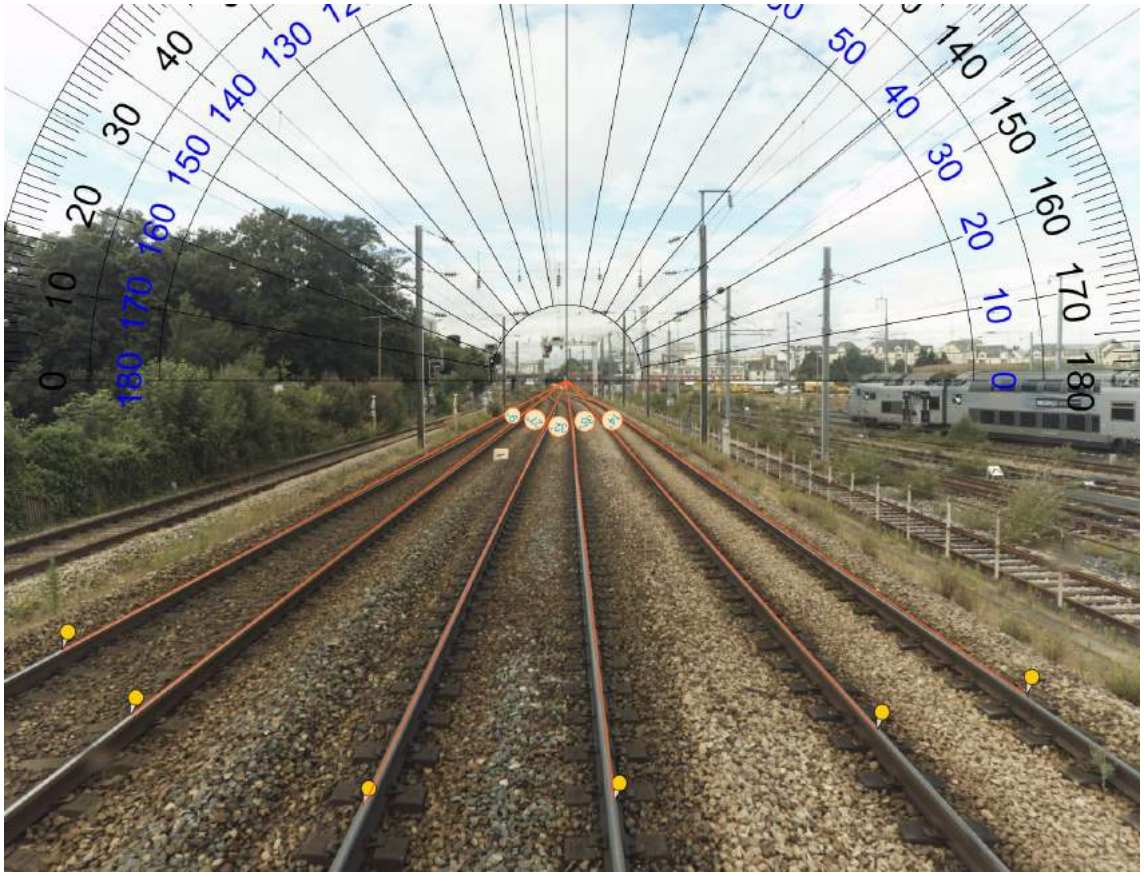


FIGURE 3.26 : Calcul des valeurs d'angle entre les rails de chaque voie à l'aide d'un rapporteur centré au niveau du point de fuite.

La deuxième idée consiste à projeter les lignes en fonction des pixels. Par exemple, une première droite est projetée à partir du point de fuite jusqu'au bas de l'image, qui correspond à y=720. Le dernier point de cette droite, lorsque y=720, est le point P. A partir du point P, on décale de quelques pixels à droite ou à gauche de P jusqu'à un point nommé P'. On trace ensuite la droite passant par le point P' et le point de fuite : c'est la nouvelle ligne projetée. On répète cette procédure pour obtenir des nouvelles droites. Afin d'affiner le travail de recherche de lignes correspondant à des rails, le pas doit être le plus petit possible : c'est le nombre de pixels à décaler entre P et P'. Si l'on applique un cercle ayant pour centre le point de fuite, l'horizon de l'image passant par le centre est un diamètre du cercle. En allant de gauche à droite, ce diamètre va d'un point A à un point B. Au point A, l'angle est de 180 degrés et au point B, l'angle est de 360 degrés. Comme on considère que les rails ne peuvent pas être des segments horizontaux, les lignes projetées ont donc des valeurs d'angles comprises entre 200 et 340 degrés. Pour pouvoir se situer pendant l'analyse, les angles sont utilisés comme référence. La projection de lignes ne se fait pas au niveau de la voie principale car on ne veut rien détecter entre les deux rails de cette voie.
Lorsqu'une droite est projetée, pour savoir si elle correspond à un rail, il faut donc analyser les deltas colorimétriques expliqués précédemment. On isole les images sous teintes de gris pour ne travailler que sur un seul canal et pouvoir calculer et analyser plus facilement les deltas de pixels. Ainsi, pour une droite, on récupère une liste de deltas et on calcule la moyenne de cette liste. Une fonction permet d'appliquer ces étapes : elle prend en entrée une image et elle retourne une liste de valeurs correspondant aux moyennes des deltas de chaque droite projetée. Il est possible de visualiser les moyennes sous forme graphique.

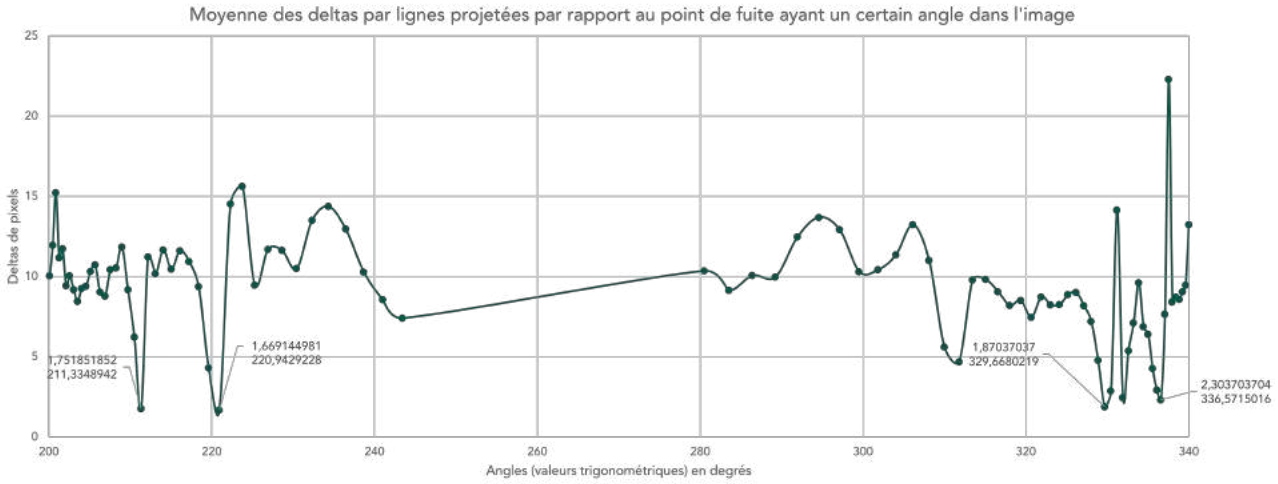


FIGURE 3.27 : Graphique représentant les deltas moyens de chaque ligne projetée dans une image en fonction de l'angle en degrés que fait la ligne (dans un cercle ayant pour centre le point de fuite).

On obtient alors un graphique correspondant à une image. Il est ensuite analysé afin de connaître les caractéristiques qui font qu'une droite est un rail. Sur ces graphiques, il est possible de voir des minima locaux : cela correspond aux valeurs les plus faibles des deltas colorimétriques des droites. Un minimum local signifie que la droite étant analysée et ayant cette valeur est en fait un rail. Sur la partie gauche de la figure 3.27, il y a deux minima locaux. L'abscisse du graphique est la valeur des angles de chaque droite projetée. Cela nous permet de repérer la droite dans l'image. Les deux minima locaux ont pour valeur d'angle environ 211 et 220 degrés : ce sont les rails de la voie parallèle à la voie principale à gauche de l'image. Sur le graphique, il n'y a pas de points entre 240 et 280 degrés : c'est parce qu'aucune droite n'y est projetée car c'est là où se situent les rails de la voie principale. Sur la partie droite du graphique, il y a plusieurs minima locaux. C'est donc important d'établir un seuil pour déterminer les valeurs à partir desquelles on considère qu'un minimum local est retenu. C'est le cas si une droite coïncide avec un mur, un quai de gare ou des traits de couleur uniforme par exemple. Des missions de 500 images sont extraites du Carrousel afin de trouver le seuil le plus adapté à toutes les situations. Cela permet aussi de lancer l'algorithme sur un plus grand nombre d'images pour voir si l'optimisation est correcte. D'après une analyse empirique, on fixe le seuil égal à 2.

Chapitre 4
Résultats et limites

Dans cette partie, nous allons voir les résultats de la méthode retenue MorphoMat en mettant en relief les avantages et les limites de celle-ci.

Tout d'abord, une fonction est créée englobant toutes les étapes expliquées dans la partie 3.2. Elles sont résumées ci-après et reprennent les méthodes décrites dans la section 3.2 :

- La fonction prend en entrée une vidéo
- Elle récupère la position des rails de la voie principale sur toute la vidéo (figure 4.1)
- Elle parcourt les images une par une et applique l'analyse des deltas colorimétriques afin de détecter les rails (de la projection des lignes par rapport au point de fuite jusqu'à l'extraction des minima locaux) (figure 4.2)
- Une fois que toutes les images sont traitées, les métadonnées des rails sont écrites dans un dataframe (sous format texte) et les images sur lesquelles les rails sont marqués en couleur sont enregistrées.

```python
def rails(video_path):

    """
        Fonction qui va faire toutes les étapes permettant pour une séquence d'images
        de retrouver les rails sur chaque image

    """

    L_df = []

    #### 1 ERE ETAPE : RECUPERATION DES VOIES PRINCIPALES DE TOUTES LES IMAGES POUR
    AVOIR LE POINT DE FUITE

    # --------------- DETECTION DES RAILS DE LA VOIE P ------------------
    coord_voie_p = calcul_voie_p.detect_voie_p_2(video_path)
    # --------------- FIN DETECTION DES RAILS DE LA VOIE P --------------------

    ### 2 EME ETAPE : PARCOURT LES IMAGES DE LA SEQUENCE

    #### 2.1 : CALCUL LES DELTAS
    #### 2.2 : CALCUL LES MINIMA LOCAUX
    #### 2.3 : RECUPERE LES BONS RAILS AVEC SAUVEGARDE DE L IMAGE

    images = os.listdir(video_path)
    for img in tqdm(images) :

        img_path = video_path + "\\" + img
```

Figure 4.1 : La première étape consiste uniquement à détecter les rails de la voie principale sur toute la vidéo. (extrait du code)

```
### 2 EME ETAPE : PARCOURT LES IMAGES DE LA SEQUENCE

#### 2.1 : CALCUL LES DELTAS
#### 2.2 : CALCUL LES MINIMA LOCAUX
#### 2.3 : RECUPERE LES BONS RAILS AVEC SAUVEGARDE DE L IMAGE

images = os.listdir(video_path)
for img in tqdm(images) :

    img_path = video_path + "\\" + img

    # ------------------------ DETECTION DES RAILS DANS L IMAGE
    ------------------------

    dict_img_f = detection_rails.dtr_img(img_path,coord_voie_p)
    df_img = pd.DataFrame(dict_img_f)
    L_df.append(df_img)

    # ------------------------ DETECTION DES RAILS DANS L IMAGE
    ------------------------

df_mission = pd.concat(L_df,ignore_index=True)

return df_mission
```

FIGURE 4.2 : Lors de la deuxième étape, les images de la vidéo sont traitées une par une afin de détecter les rails restants en appliquant l'analyse des deltas colorimétriques. (extrait du code)

4.1 Résultats

La fonction est appliquée sur des vidéos contenant environ 500 images. En effet, cela permet de stabiliser au mieux les positions des rails de la voie principale et ainsi avoir de meilleurs résultats. Sur la figure 4.3, les rails sont correctement détectés : à quelques pixels près, les traits de couleur indiquant la présence d'un rail sont placés au bon endroit. Afin de mieux analyser nos résultats, les images extraites sont toutes issues d'environnements variés.

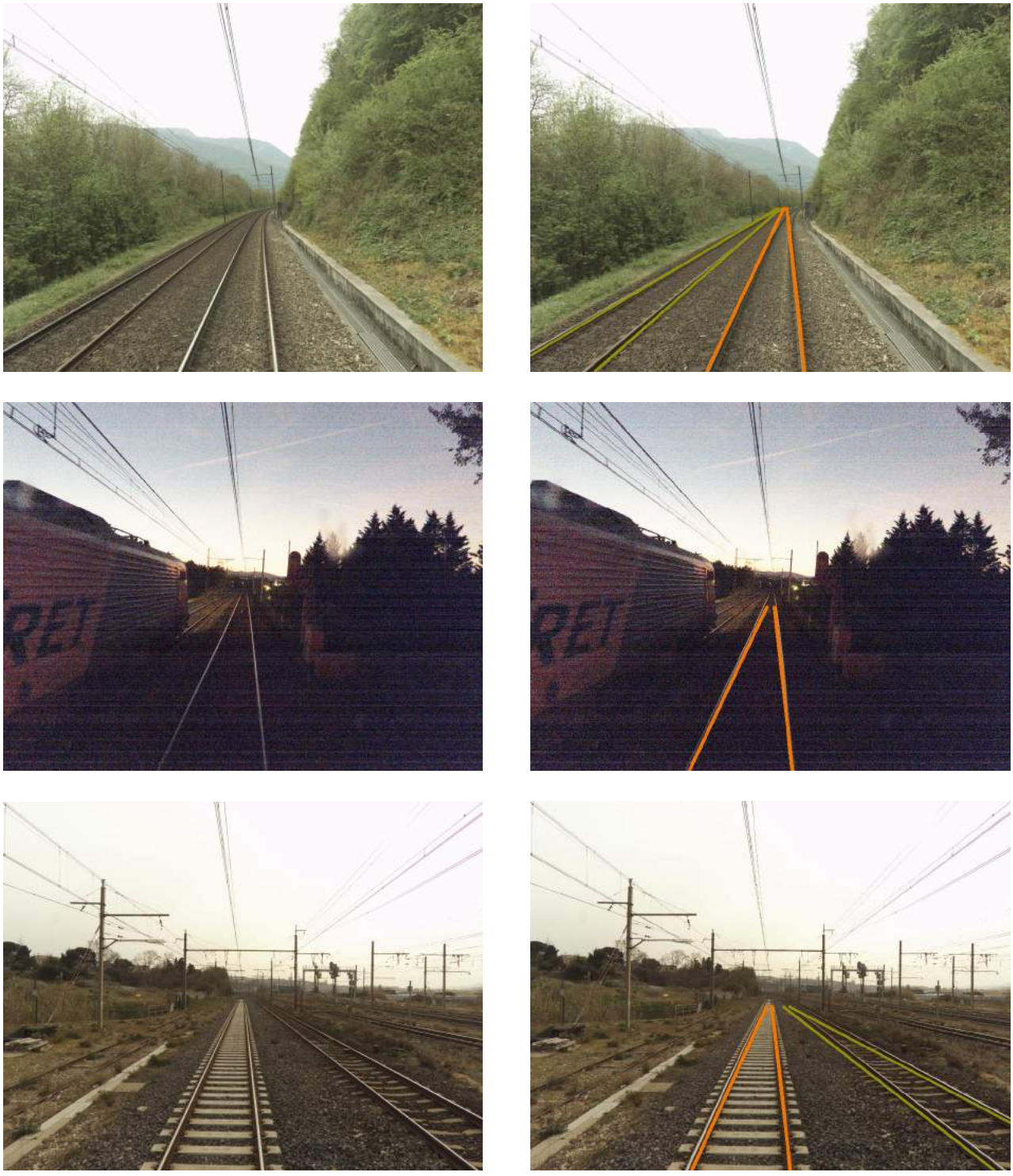


FIGURE 4.3 : Des exemples de résultats lorsque les rails sont détectés dans une image.

Les résultats sont récupérés à l'aide d'un dataframe (figure 4.4). Une vidéo a un dataframe. Celui ci est composé de 6 colonnes :

- Le premier champ correspond au nom de l'image de la vidéo. Le nom contient le numéro de la ligne, du rang, de la voie et le numéro de l'image extraite (par exemple : 890000-1-V2-0000002897 est la 2897ème image extraite de la mission où l'ESV circule sur la voie 2 au rang 1 de la ligne 890000)

- Le deuxième champ concerne le numéro du rail détecté sur l'image. Si ce sont les rails de la voie principale, alors il est indiqué *rail_vp_right* et *rail_vp_left* : ce sont les rails à droite et à gauche de la voie principale. Ainsi, pour chaque image, il y a toujours deux lignes consacrées à cette voie.
- Les paramètres de droite du rail sont inscrits dans la troisième et quatrième colonne. Ces paramètres restent inchangés pour les rails de la voie principale.
- L'avant-dernière colonne contient la valeur de l'angle trigonométrique en degré du rail.
- La valeur du delta moyen calculé lors de l'analyse des deltas colorimétriques dans la section 3.2 est enregistrée dans la dernière colonne. Pour les rails de la voie principale, cette valeur n'est pas calculée : un 0 est mis à la place.

df_2.txt - Bloc-notes

Fichier Edition Format Affichage Aide

```
890000-1-V2_0000002897 rail_vp_right -2.121212121212121 1449.4223965543763 244.7594707352132 0.0
890000-1-V2_0000002897 rail_vp_left 9.130434782608695 -4169.347826086956 276.2503395777805 0.0
890000-1-V2_0000002897 rail_1 -0.8784332644973114 828.8120189690135 221.2971476474872 2.037037037037037
890000-1-V2_0000002897 rail_2 -0.5937937445495451 686.6706886783929 210.70157560630406 2.216417910447761
890000-1-V2_0000002897 rail_3 0.8794618852013745 -49.033598306668665 318.70463461940824 2.914141767057048
890000-1-V2_0000002898 rail_vp_right -2.121212121212121 1449.4223965543763 244.7594707352132 0.0
890000-1-V2_0000002898 rail_vp_left 9.130434782608695 -4169.347826086956 276.2503395777805 0.0
890000-1-V2_0000002898 rail_1 -0.8784332644973114 828.8120189690135 221.2971476474872 2.0223048327137545
890000-1-V2_0000002898 rail_2 -0.5937937445495451 686.6706886783929 210.70157560630406 2.300380228136882
890000-1-V2_0000002898 rail_3 0.8548365237005485 -36.73635425827666 319.4829824462539 2.5524968789013736
890000-1-V2_0000002899 rail_vp_right -2.121212121212121 1449.4223965543763 244.7594707352132 0.0
890000-1-V2_0000002899 rail_vp_left 9.130434782608695 -4169.347826086956 276.2503395777805 0.0
890000-1-V2_0000002899 rail_1 -0.8784332644973114 828.8120189690135 221.2971476474872 1.9814126394052045
890000-1-V2_0000002899 rail_2 -0.5937937445495451 686.6706886783929 210.70157560630406 2.4099616858237547
890000-1-V2_0000002899 rail_3 0.8548365237005485 -36.73635425827666 319.4829824462539 2.5044275510630296
```

FIGURE 4.4 : Une partie du dataframe récupéré en sortie de la fonction permettant de détecter les rails.

4.2 Limites

FIGURE 4.5 : Les rails de la voie principale ne sont pas correctement positionnés.

La méthode n'est pas 100% optimale. Elle présente malheureusement des limites. Tout d'abord, si la première étape de MorphoMat n'est pas correctement appliquée, le point de fuite est mal situé. Si le point de fuite est mal situé, les rails des autres voies ne sont pas correctement détectés. Pour rappel, la première étape consiste à détecter les rails de la voie principale. Le grand nombre d'images sur lesquelles les rails sont détectés permet de consolider les positions de ces derniers. Cependant, en présence d'un virage, à l'horizon, les rails ne paraissent pas droits mais forment plutôt une courbe. Cela perturbe la consolidation, même si c'est avec la médiane que les paramètres de droite de ces rails sont calculés.

Ensuite, pendant la deuxième étape, les rails ne sont pas tout le temps détectés au bon nombre (figure 4.8c). Il peut aussi y avoir les deux extrêmes : soit aucun rail n'est détecté (figure 4.8b), soit trop de rails sont considérés dans l'image (figure 4.8a). En effet, sur certaines images, il y a des tendances linéaires, ce qui faussent l'analyse des deltas colorimétriques. Aussi, il se peut que le seuil établi dans la section 3.2 soit trop élevé : il y a plus de rails détectés que prévu. A contrario, si le seuil est trop bas, il peut en résulter qu'aucune ligne ne soit considéré comme étant un rail. Comme expliqué précédemment, le seuil est déterminé empiriquement et doit être le plus faible possible.

(a) Seuil = 2

(b) Seuil = 3

(c) Seuil = 4

FIGURE 4.6 : Les résultats de MorphoMat sur la même image mais avec des seuils d'analyse colorimétrique différents.

FIGURE 4.7 : Ce sont deux images consécutives mais sur la deuxième, un rail n'est pas détecté.

(a) Des rails sont détectés à droite de l'image alors que ce n'est seulement les traces linéaires sur le mur.

(b) Sauf la voie principale, aucun rail n'est détecté.

(c) Il manque des rails à détecter sur cette image.

FIGURE 4.8 : Ces images sont comprises dans la même vidéo, pourtant les résultats de la détection des rails différent.

Par ailleurs, cela rend la méthode irrégulière sur ses résultats. Par exemple, sur une image, les rails peuvent être correctement détectés. Or, sur l'image consécutive, sur laquelle il y a le même

nombre de rails que la précédente, aucun rail n'est détecté.
De plus, MorphoMat restreint le travail de détection des voies. Elle ne permet d'obtenir uniquement les rails de la voie principale et les rails parallèles à cette voie. Tous les rails faisant l'objet d'une intersection ne sont pas pris en compte. En outre, la méthode s'appuie sur les propriétés mathématiques des fonction affines. Comme vu précédemment, les rails ne forment pas toujours des lignes droites. Or, il est considéré ici qu'une grande partie du rail forme une droite dans l'image donc l'idée de travailler avec une forme géométrique est mise de côté. Enfin, le temps que la méthode met pour traiter une vidéo contenant 500 images est d'environ 3 minutes. Une vidéo contient 2500 images. Le temps de traitement pour une vidéo est donc de 15 minutes.

Conclusion et perspectives

4.3 Conclusion

Pour conclure, la méthode retenue pour détecter les voies dans les images est la méthode dite **MorphoMat**. Elle se repose principalement sur les algorithmes de Canny et de Hough mais aussi sur l'analyse plus fine des pixels sur les images. L'utilisation du Deep Learning n'est pas adaptée aux détection de lignes dans les images de la base terrain : elle nécessite un temps de correction important. MorphoMat permet de fiabiliser les rails de la voie sur laquelle l'ESV circule (la voie principale), ce qui permet de détecter déjà une des voies des images. Elle permet aussi de détecter uniquement les rails des voies parallèles à la voie principale. Cependant, elle présente des irrégularités : certains rails ne sont pas reconnus ou trop de lignes sont considérées comme étant des rails. c'est pourquoi il est nécessaire, par la suite, de réfléchir à l'optimisation de MorphoMat.

FIGURE 4.9 : Résultat de la méthode MorphoMat sur une image d'un ESV.

4.4 Perspectives

Dans le cadre d'amélioration de MorphoMat, il est possible de s'intéresser aussi aux rails des autres voies : celles qui ne sont pas parallèles à la voie principale. Or, avant d'élargir la détection des rails sur les autres voies, il est nécessaire de s'assurer que la méthode fonctionne avec une précision suffisante. En effet, les résultats de la méthode montrent que les rails ne sont pas détectés de manière réglulière

sur toutes les images d'une vidéo. L'idée est donc de réfléchir à une manière de consolider les rails entre les images. Pour se faire, il faut mettre en place des hypothèses. Par exemple, si les mêmes rails sont présents dans l'image sur une série d'images consécutives, alors leurs positions sont fixés. Il faut donc penser au nombre d'images à la suite pour considérer l'hypothèse correcte.
Comme l'objectif est de détecter les voies et que l'on connaît l'écart entre les rails d'une même voie (1435 mm), il est possible de trouver une fonction permettant de récupérer les voies sur les images à partir des rails détectés. Dans les métadonnées que le fournisseur ImajNet donne, il y a des informations de la caméra notamment la focale.
L'équipe I2D a aussi d'autres projets liés à la détection des voies, notamment le changement de référentiel expliqué dans la sous-section 0.0.3. Par ailleurs, la position des objets dans l'image est calculée par rapport à la voie. En effet, le fil de rail peut être déterminé. Il s'agit de la ligne entre deux rails d'une voie : c'est aussi l'abscisse curviligne. A partir de cet axe, par projection orthogonale, il est possible de connaître la distance entre un objet et le fil de rail.
Cependant, la priorité est de corriger la donnée et donc d'utiliser la détection des voies pour aider la détection des objets.

Bibliographie

Dooze, D., Khoudour, L., & Vieren, C. (1998). Obstacle detection in front of automatic trains by linear stereo vision. *WIT Transactions on The Built Environment*, *37*.

Barnes, J., Rizos, C., Wang, J., Small, D., Voigt, G., & Gambale, N. (2003). High Precision Indoor and Outdoor Positioning using *LocataNet*. *Journal of Global Positioning Systems*, *Vol 2*(2), 73-82.

Dach, R., Hugentobler, U., Fridez, P., & Meindl, M. (2003). *Bernese GPS Software Version 5.0*. Astronomical Institute, University of Bern. Bern, Switzerland.

Samama, N. (2008). *Global positionning : Technologies and Performance*. John Wiley & Sons, Inc.

Jardak, N., & Samama, N. (2010). Short Multipath Insensitive Code Loop Discriminator. *IEEE*.

Vervisch-Picois, A. (2010). *Etude de Systèmes de Positionnement en Intérieur Utilisant des Mesures de Phase du Code ou de Phase de la Porteuse de Signaux de Navigation par Satellites* (thèse de doct.). Telecom SudParis.

Bosser, P. (2011). *GNSS : Systèmes Globaux de Positionnement par Satellite* (rapp. tech.). ENSG/IGN.

Schmitt, M., & Mattioli, J. (2013). *Morphologie mathématique*. Presses des MINES.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, 2961-2969.

Kalms, L., Rettkowski, J., Hamme, M., & Göhringer, D. (2017). Robust lane recognition for autonomous driving. *2017 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 1-6.

Wang, Y., Wang, L., Hu, Y. H., & Qiu, J. (2019). RailNet : A Segmentation Network for Railroad Detection. *IEEE Access*, *7*, 143772-143779. https://doi.org/10.1109/ACCESS.2019.2945633

Ye, T., Zhang, Z., Zhang, X., & Zhou, F. (2020). Autonomous Railway Traffic Object Detection Using Feature-Enhanced Single-Shot Detector. *IEEE Access*, *8*, 145182-145193. https://doi.org/10.1109/ACCESS.2020.3015251

Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Gläser, C., Timm, F., Wiesbeck, W., & Dietmayer, K. (2021). Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving : Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems*, *22*(3), 1341-1360. https://doi.org/10.1109/TITS.2020.2972974

Ye, T., Zhang, X., Zhang, Y., & Liu, J. (2021). Railway Traffic Object Detection Using Differential Feature Fusion Convolution Neural Network. *IEEE Transactions on Intelligent Transportation Systems*, *22*(3), 1375-1387. https://doi.org/10.1109/TITS.2020.2969993

Li, X., & Peng, X. (2022). Rail Detection : An Efficient Row-Based Network and a New Benchmark. *Proceedings of the 30th ACM International Conference on Multimedia*, 6455-6463. https://doi.org/10.1145/3503161.3548050

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023). Segment Anything.

Zhang, Y., Li, K., Zhang, G., Zhu, Z., & Wang, P. (2023). DFA-UNet : Efficient Railroad Image Segmentation. *Applied Sciences*, *13*(1). https://doi.org/10.3390/app13010662

Barnes, J., Rizos, C., Wang, J., Small, D., Voigt, G., & Gambale, N. (s. d.). *Locata : A New Postioning Technology for High Precision Indoor and Outdoor Positioning*.

Petrovski, I., Okano, K., Kawaguchi, S., Torimoto, H., Suzuki, K., Toda, M., & Akita, J. (s. d.). Indoor Code and Carrier Phase Positioning with Pseudolites and Multiple GPS Repeaters.

Vervisch-Picois, A., Samama, N., Jardak, N., & Fluerasu, A. (s. d.). GNSS Repeater Based Approach for Indoor Positioning : Current Status.

Kee, C., Yun, D., Jun, H., Parkinson, B., Pullen, S., & Lagenstein, T. (2001). Centimeter-accuracy indoor navigation Using GPS-Like Pseudolites. *GPS World*.

Kee, C., & Yun, D. (2009). *US 2009/0002230 A1 : Pseudolite-based precise positioning system with synchronised pseudolites* (rapp. tech.). US Patent.

Table des figures

Liste des tableaux

Annexes

Jeu de données pour les modèles de Deep Learning

Annexe A

Figure A.1 : Quelques images du jeu de données qui sont classées comme "Beau Temps". A l'horizon, les rails sont soit un peu courbés, soit droits.

FIGURE A.2 : Quelques images du jeu de données qui sont classées comme "Mauvais Temps" et "Avec Obstacles". A l'horizon, les rails sont soit un peu courbés, soit droits.

Travail sur RailNet

Annexe B

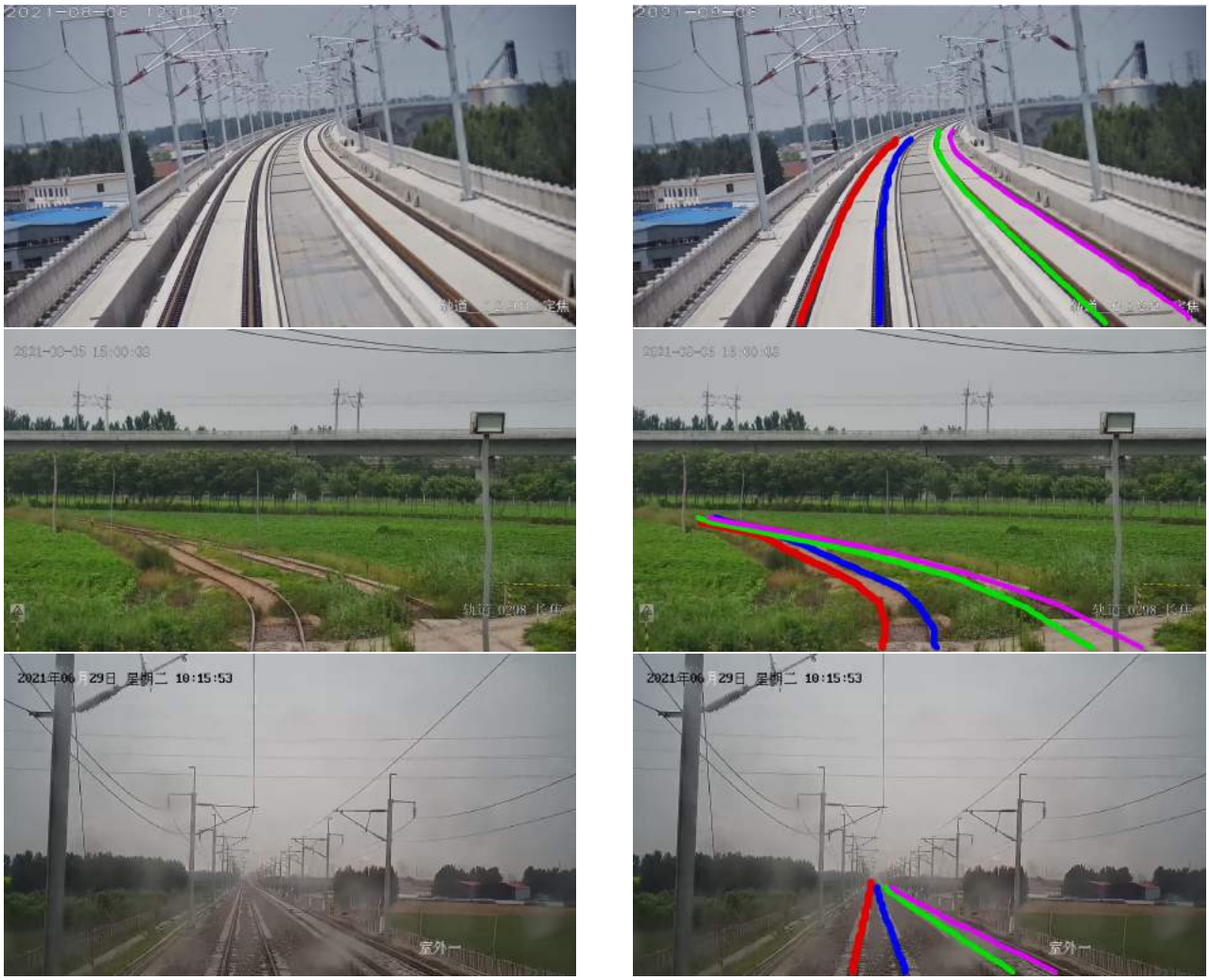


Figure B.1 : Les résultats de RailNet sur trois images de RailDB (à droite, les lignes de couleur superposent correctement les rails).

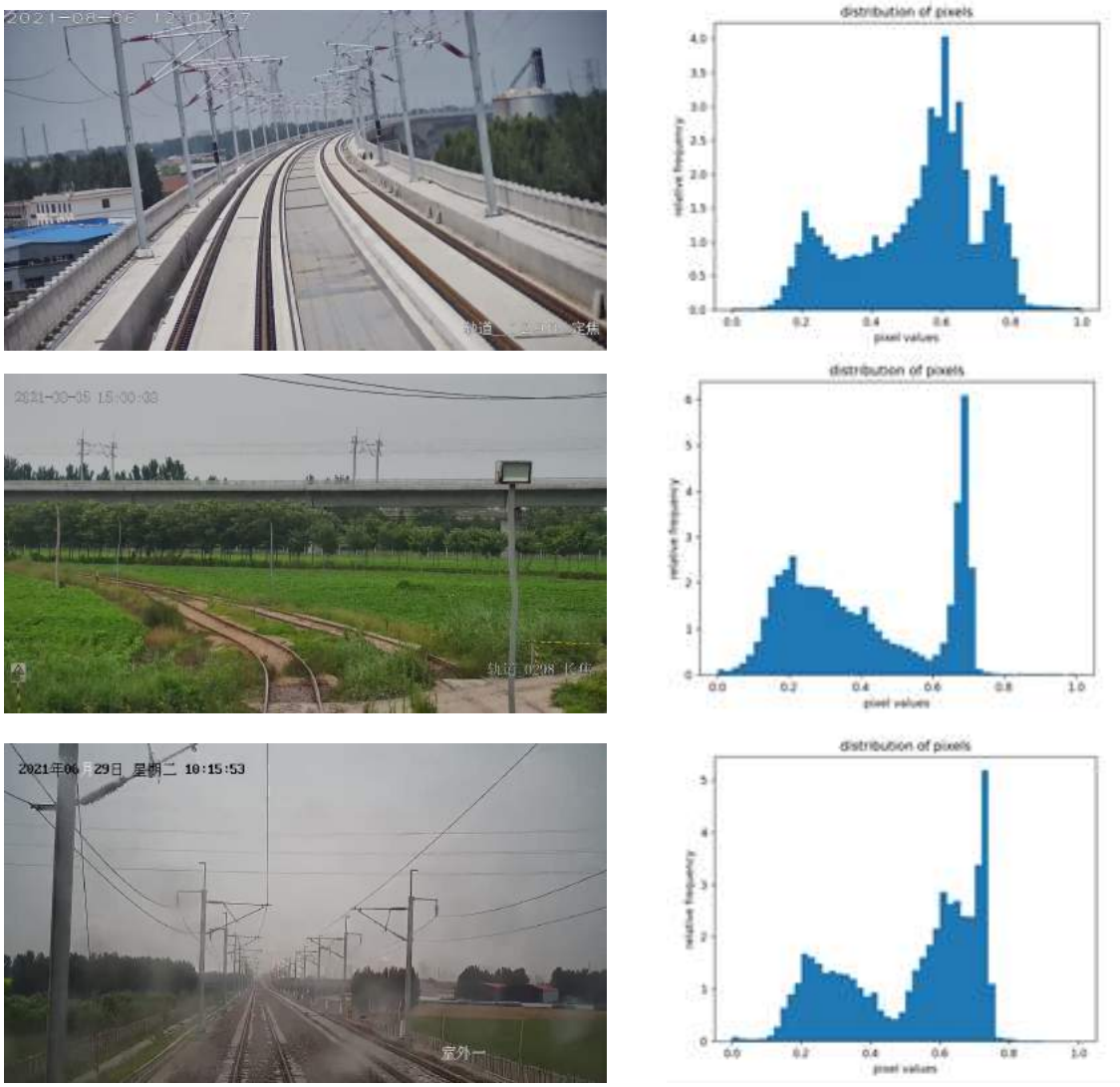


FIGURE B.2 : Les histogrammes de trois images de RailDB.

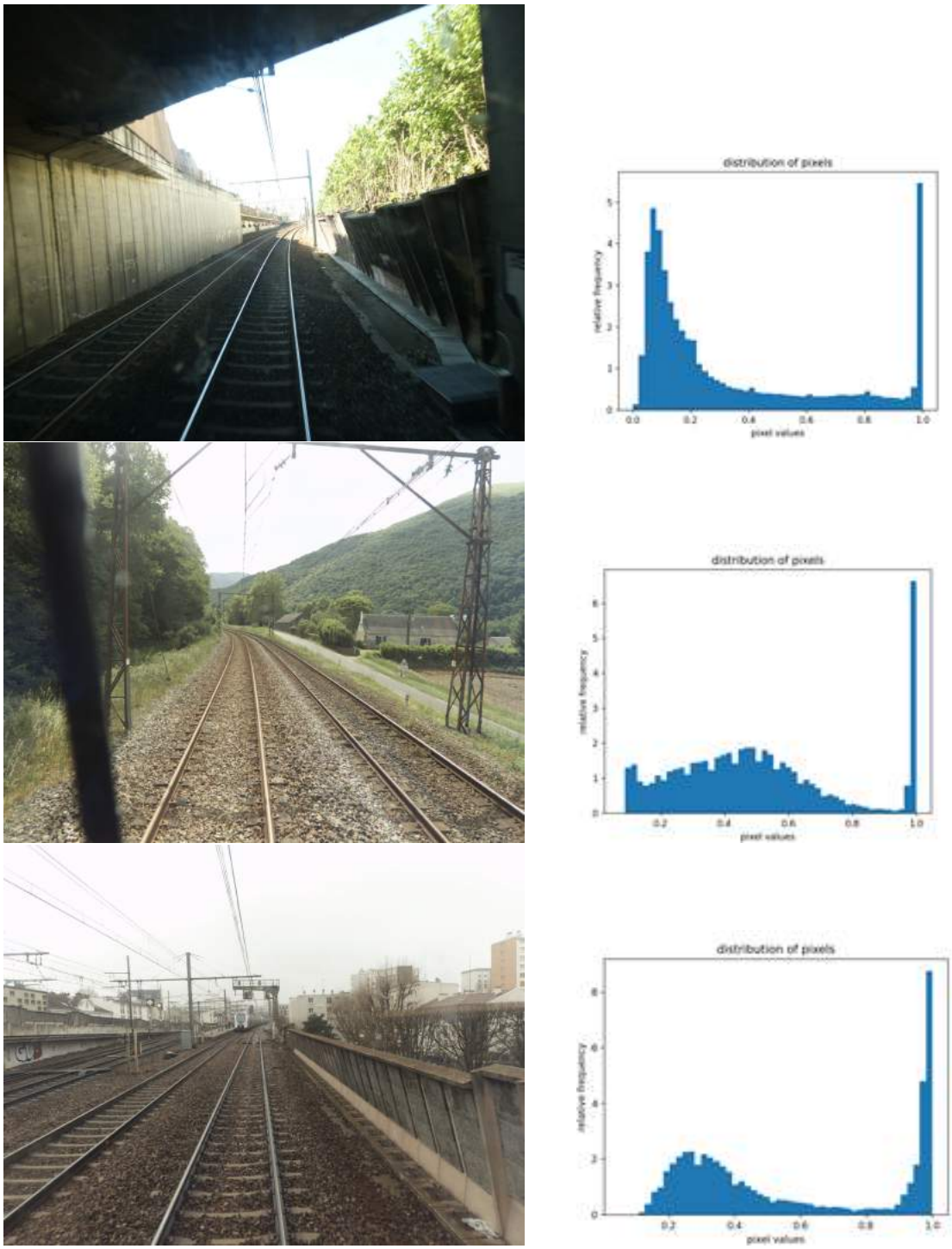


FIGURE B.3 : Les histogrammes de trois images des ESV.