Le génie pour l'industrie

Projet final

Par

David Olivier OLID28049706

Soumis le 21 avril 2020

Cours : ELE 747

Trimestre Hiver 2020

Introduction

Le projet détaillé dans ce rapport, comme son titre l'indique "**Détection de tronc d'arbres par analyse d'images**" vise à développer la base d'un système qui pourrait ultimement permettre à des véhicules autonomes de naviguer dans un milieu forestier. Plus concrètement, l'algorithme conçu permet d'identifier les troncs d'arbres, à proximité de la caméra, dans des images prises en forêt.

Par ailleurs, afin d'assurer que le projet conserve son aspect pratique, certaines des images utilisées pour développer le système ont été capturées par un drone (DJI Tello). Le but était d'assurer que le thème de la navigation autonome sois pris en compte. Cependant, la contrainte qui a le plus guider le développement est la nécessité d'utiliser des techniques de traitement d'images qui sont pertinentes au cours d'ELE739.

Objectifs et observations

Comme mentionner précédemment, l'objectif principal est de réaliser une segmentation des images prises en milieu forestier. Plus précisément, on cherche à identifier les pixels qui font partie de tronc d'arbres situés près de la caméra.

Ainsi, on cherche à classifier chaque pixel de l'image comme membre d'une de ces deux classes :

- Pixel de tronc d'arbre
- Pixel de fond

Consulter la figure No.1 pour voir des exemples d'images à segmenter.
Consulter la figure No.2 pour voir des exemples de la segmentation espérée.

On remarque que l'un des défis en lien avec les images choisies est que l'on doit être en mesure d'ignorer les arbres plus lointains (arbres assez distants pour être considérés comme faisant partie du fond). De plus, certains arbres situés à une distance moyenne sont plus difficiles à distinguer à cause de la texture des arbres plus lointains qui leur est très semblable.

On observe également que pour toutes les images, le sol est couvert de neige puisqu'elles ont été capturées en hiver. On retrouve donc un grand contraste entre le sol et les troncs d'arbres, chose qui risque de faciliter les classifications. Cependant, nous allons voir plus tard dans ce rapport que la présence de cette neige apporte également des effets non désirables.

Travaux connexes

La première étape dans le développement a été la recherche d'un article scientifique décrivant un projet similaire à celui-ci. Le but étant de trouver des approches ou des techniques pré- existantes pour réaliser un système de segmentation semblable.

L'article sur lequel se base ce projet s'intitule "**Visual tree detection for autonomous navigation in forest environment**". Il a été écrit en 2008 par deux chercheurs de l'université de Umeå en Suède.

L'article décrit un système de traitement d'image connecté à une caméra fixée sur le côté d'un tracteur. Ce système est en mesure d'identifier les troncs d'arbres à proximité du tracteur et d'estimer leur distance. Pour mon projet, seule la partie de cet article traitant de la segmentation est pertinente et le reste sera donc mis de côté.

Consulter la figure No.3 pour voir le projet de l'article dans son ensemble.

On note que les auteurs n'ont pas partager les images qu'ils ont utilisées pour la conception. Ainsi, l'approche qu'ils proposent sera implémentée, modifiée et une étude des résultats sera conduite sur les images que j'ai capturées. L'approche générale est les modifications apportées seront traités dans la prochaine section.

En regardant les images utilisées dans l'article on remarque quelques faits pertinents. Premièrement, les images ont été capturées avec une caméra inclinée vers le bas, ce qui fait que l'on retrouve peu de troncs d'arbres et autres objets en l'arrière-plan. Les images du projet actuel ont donc un arrière-plan plus bruité et on peut s'attendre à ce que la segmentation soit plus complexe. Deuxièmement, on note que la couverture du sol varie dans images de l'article car elles ont été prises à différentes saisons. Il y a donc plus de variations de texture et de couleur entre les images de l'article qu'entre celles de ce projet. Ceci simplifie le projet actuel, mais le système final sera moins robuste car il pourra uniquement fonctionner en hiver.

Consulter la figure No.4 pour voir des exemples d'images à segmenter (dans l'article).
Consulter la figure No.5 pour voir un exemple de segmentation (dans l'article).

Paramètres	Projet (2020)	Projet (article - 2008)
Objectif	1) Détection de troncs d'arbres à proximité de la caméra.	1) Détection de tronc d'arbres. 2) Estimation de la distance entre la caméra et le tronc.
Nombre d'images disponibles	42 images 1246 segments de fond. 551 segments de tronc.	Inconnu
Classes pour la segmentation	1) Segment de fond 2) Segment de tronc	1) Segment de fond 2) Segment de tronc
Descriptions des images	Arrière-plans bruités par plusieurs troncs lointains. Images prises uniquement en conditions hivernales.	Arrière-plans moins bruités. Images prises pendant les 4 saisons (plus de couleurs et textures).

Le tableau ci-haut rassemble les comparaisons de haut niveau qui peuvent être faites par rapport aux deux projets.

Approche globale

Consulter la figure No.6 pour voir un schéma représentant l'approche générale.

Dans un premier temps, l'image fournie à l'entrée du système est redimensionnée et divisée en plusieurs sous-images. Cette étape est directement empruntée de l'article de 2008 et sert à réduire la quantité de classifications nécessaires pour identifier les troncs d'arbres. Ainsi, le système analyse et classifie chaque sous-image de façon atomique et l'appartenance à la classe choisie est transmise à leur pixels

constituants. À l'entrée, les image passent de 16M pixels à (320 x 240) pixels pour ensuite être divisées en 192 sous-images de dimensions (20 x 20) pixels.

Après la division, les sous-images passent au travers un pipeline d'extraction de descripteurs de couleur et de texture. Les descripteurs choisis seront couverts en détails dans la prochaine section, pour l'instant nous allons considérer leur ensemble comme un simple vecteur de données. Ainsi, chaque sous-image est décrit par un vecteur de données qui est utilisé pour performer une classification. La classification est faite avec une Model de type SVM (Support Vector Machine) et l'approche d'entrainement sera visité plus en détails dans une autre section.

Une fois que tous les sous-images ont été classifiées, on les assemble et, naturellement, celles appartenant à la classe "tronc" s'alignent de sorte que les troncs d'arbres soient segmentés. Le système retourne une image redimensionnée où tous les divisions (sous-images) ayant été classifiés somme "tronc" sont encadrés en couleur.

Extraction de données

Cette section détaille la théorie et l'implémentation de l'extraction de descripteurs pour quantifier les couleurs et les textures. Les trois descripteurs choisis pour ce projet correspondent à ceux qui ont engendrés les meilleurs résultats parmi les 8 descripteurs explorés dans l'article de 2008.

On note que pour chaque descripteur extrait, une normalisation a été faite sur son vecteur de données afin que chacun de ses membres soit dans la plage de valeurs [0, 1]. En général, les modèles de classification, le SVM entre autres, se comportent mieux quand on leur fournit des données normalisées. Si, lors de l'entrainement, on utilise des descripteurs qui ont des plages de valeurs relatives différentes, on court le risque que les descripteurs centrés sur de plus petites valeurs soient ignorés. Autrement dit, normaliser les descripteurs fait en sorte qu'ils sont considérés de façons égale lors de l'entrainement

1. Descripteurs de couleur

Les descripteurs de couleurs ont pour but de quantifier la présence et la répartition des couleurs dans une image de façon efficace. Nous allons voir que dans ce pipeline, les descripteurs de couleurs forment un vecteur de 51 données.
Il faut noter que les images qui entrent dans le pipeline d'extraction sont dans le domaine de couleur RGB. Tout comme dans l'article de 2008 le domaine HSI a été exploré mais pour ce projet, c'est le domaine RGB qui permet d'atteindre des meilleurs résultats.

1.1 Histogrammes

Pour chacun des canaux de l'image fournie (R, G, B) on calcul un histogramme de 15 bandes. Par la suite, on concatène les 3 histogrammes de sorte à obtenir un vecteur de 45 données. Comme mentionné précédemment, ce vecteur doit être normalisé et pour le faire on divise les valeurs de chaque histogramme par la quantité de pixels dans l'image.

La figure No.7 montre l'allure d'un vecteur de données d'histogramme.

1.2 Statistiques

Pour chacun des canaux (R, G, B) de l'image on calcule la valeur moyenne est l'écart type de la distribution de valeurs de pixels. On pourrait argumenter que cette information est déjà présente dans le descripteur de d'histogramme de couleur mais, en pratique, l'inclusion de ce descripteur améliore les résultats de classification.

Ainsi, pour chaque canal de couleur, deux valeurs sont calculées pour former un vecteur de 6 valeurs. On note que pour normaliser le vecteur, on divise tous ses membres par le nombre 255 qui correspond à la plus grande intensité qu'un pixel peut avoir. Ce cette façon on est certain que les valeurs dans le vecteur de données seront entre 0 et 1.

La figure No.7 montre l'allure d'un vecteur de données de statistiques.

2. Descripteurs de texture

La première chose à noter est que le descripteur de texture utilise la version "GrayScale" des sous-image.

Le pipeline d'extraction utilise un seul descripteur de texture soit le LBP (Local Binary Pattern). Ultimement, ce descripteur permet d'assigner une signature (vecteur de nombres) à chaque type de texture présente dans une image de sorte que des segments possédant des textures similaires produisent des signatures similaires. Ainsi, il suffit ensuite de comparer les signatures générées pour pouvoir différencier différentes textures.

Les figures No.8 et No.9 montrent les signatures associées à différentes textures.

L'idée d'utiliser ce descripteur provient entièrement de l'article de 2008 où la version classique du LBP est appliquée sur les sous-images. Pour ce projet, j'ai choisi d'implémenter une variation de cette version qui permet d'obtenir une signature plus courte pour représenter les différentes textures.

2.1 Implémentation

Premièrement, dans un processus séquentiel qui ressemble beaucoup à une convolution, on seuille les pixels positionnés sur le cercle de voisinage de chaque pixel. On note que le la distance (rayon) utilisée pour déterminer le voisinage peut varier.

La figure No.10 illustre le processus de seuillage ainsi que différents cercles de voisinage.

Après le seuillage, on forme une chaîne binaire en déroulant le cercle de voisinage seuillé. Le patron binaire de cette chaîne est ensuite classifié comme faisant partie d'un des (N+2) patrons binaires possibles, où N est le nombre de voisins. On note que ces (N+2) patrons possibles sont constituées par les (N+1) patrons uniformes et que tous les autres patrons non uniformes sont mis dans la même classe. Chaque patron uniforme peut être associé à un type de variation de texture (**voir figure No.11**).

On définit les patrons binaires uniformes et non uniformes selon la règle suivante :

- Les patrons uniformes contiennent jusqu'à une variation (0 -> 1) et jusqu'à une variation (1 -> 0).
- Tous les patrons qui ne sont pas uniformes sont non uniformes.

Les figures No.11 et No.12 montrent quelques exemples de patrons uniformes/non uniformes ainsi qu'un exemple complet de classification de patrons pour un chaîne binaire de longueur 4.

Une fois que chaque pixel de l'image est associé à un des (N+2) classes, on génère l'histogramme de cette nouvelle image. On note que cet histogramme contiendra (N+2) bandes et qu'il correspondra à la signature de texture de l'image. De plus, comme pour les autres descripteurs, le vecteur de données correspondant à l'histogramme est normalisé en divisant tous ses membres par le nombre total de pixel de l'image.

La figure No.13 montre l'allure d'un vecteur de données de texture.

Il est important de noter que même si le descripteur LBP, compte les différentes directions de variation d'intensité, il reste insensible aux amplitudes des variations. En effet, une variation d'intensité de 40 sera interprétée de la même façon qu'une variation de 1. Cette caractéristique fait en sorte que des patrons de texture non prévisibles peuvent être retrouvée sur des surfaces qui semblent très uniforme.

La figure No.14 montre image LBP résultant pour un fond de neige.

Fusion et normalisation

Toujours en se fiant à la figure No.6, on voit que l'étape suivant l'extraction est la fusion des descripteurs. L'objectif de cette étape est d'assembler les descripteurs afin de former un seul vecteur de données contenant l'information relative aux couleurs et aux textures. Ultimement, c'est ce vecteur qui sera utiliser par le modèle pour performer une classification. Pour accomplir cet objectif, l'article de 2008 couvre deux méthodes de fusion distinctes.

Consulter les figure No.15 et No.16 pour voir les deux méthodes de fusion des descripteurs.

En pratique, on constate que choisir l'une ou l'autre des deux méthodes n'a pas un grand impact sur la performance du système final (observé par gridsearch). En effet, on voit que les deux méthodes assemblent les descripteurs de couleur de la même façon et que la différence se trouve dans la manière que le LBP est calculé. Ainsi, la méthode No.1 a été sélectionnée puisqu'elle génère un vecteur de données plus court (63 membres) et qu'elle nécessite moins de calcul que l'autre méthode (un seul calcul de LBP).

La fusion précède une dernière étape de normalisation où les membres du vecteur de données (63 membres) sont ajustés de sorte que la norme soit de 1. Le but est d'assurer que tous les différents descripteurs partagent la même plage de valeurs, soit la plage [0, 1].

La figure No.17 montre l'impact sur les résultats quand on retire les étapes de normalisation.

Classification et mesure de performance

Le fonctionnement d'un modèle de classification (SVM) sort un peu du cadre du cours, ainsi cette section va se concentrer sur la mesure de performance utilisée ainsi que la méthode de recherche par grille "GridSearch".

Mesure de performance

En termes de mesure de performance, on cherche à utiliser une mesure qui reflète bien la précision du modèle de classification. Quand on regarde les proportions de la base de données d'entrainement, on remarque qu'il y a beaucoup plus sous-images ayant une étiquette « fond » (1246) qu'il y en a avec une étiquette « tronc » (551). Ainsi, il est important de choisir une mesure qui ne sera pas biaisée par ces proportions inégales. J'ai donc choisi d'utiliser le « recall » pour quantifier la performance car, pour le contexte du projet, c'est une mesure qui s'intéresse seulement aux données possédant l'étiquette « tronc ». En effet, les données avec l'étiquette « fond » seront ignorées, ce qui fait du sens pour cette application puisqu'on priorise la détection de troncs d'arbres.

La figure No.18 montre la formule de précision classique et la formule de "recall".
La figure No.19 montre le comportement des deux mesures de performance pour des données d'entrainement avec proportions inégales.

On remarque que dans le scénario où le modèle rate toutes les classifications de tronc d'arbres, la mesure de précision classique retourne une note d'environ 70%. On voit donc que cette mesure n'est aucunement représentative de la capacité du modèle à reconnaitre des troncs d'arbres.

Recherche par grille "GridSearch"

Les trois méthodes d'extraction de descripteurs ainsi que le modèle SVM nécessitent qu'on leur fournisse des paramètres de configuration relatifs à leurs fonctionnements. Par exemple, l'extraction du descripteur LBP nécessite que l'on définisse un rayon de voisinage ainsi que le nombre de voisins à considérer. En général, la variation de chaque paramètre affectera la performance globale du système. Puisqu'on veut maximiser la note de performance obtenue, on se retrouve avec un problème d'optimisation où on cherche le "recall" maximal en fonction des paramètres de configuration.

Une des façons de résoudre ce problème est d'effectuer une recherche par grille. Cette méthode implique que l'on va essayer plusieurs agencements de valeurs de paramètres afin de trouver ceux qui apportent les meilleurs résultats. Ainsi, le prérequis de cette approche est de définir une plage discrète de valeurs pour chaque paramètre à considérer. L'ensemble des plages de valeurs est défini comme étant une grille. Par la suite, on essaye tous les agencements possibles dans la grille en prenant soins de noter la performance associée à chacun d'entre eux. Finalement, on sélectionne l'agencement qui a obtenu la meilleur note (recall).

On note que la grille de la figure 21 possède 432 agencements possibles et que par conséquent, la recherche est un processus très lent qui peut prendre des heures à compléter.

La figure No.20 montre la grille de valeurs de paramètres utilisée pour la recherche par grille.
La figure No.21 montre les valeurs de paramètres retenues après la recherche.

Résultats

Le système de classification obtient une note de “recall” moyenne de 83.82 %. Ceci signifie que le système est en mesure de d’identifier correctement 83.82% des étiquettes de l’ensemble de données d’entrainement.

La figure No.22 montre la performance du système face à des images qui ne sont pas dans l’ensemble d’entrainement.

Analyse

Globalement, le système performe assez bien, mais il comporte des faiblesses notables que je vais analyser dans cette section.

Premièrement, on voit que le système éprouve de la difficulté à détecter les troncs d’arbres de couleur plus claire. La cause principale est le fait que l’ensemble des images d’entrainement ne comprend pas beaucoup de ce type de troncs. Si on veut que le système se généralise mieux, il faudrait retourner en forêt pour capturer plus d’exemples de tronc d’arbres claires.

Deuxièmement, on remarque qu’à plusieurs occasions le système interprète des blocs de neige comme étant des segments de troncs d’arbres. D’ailleurs, ce sont les erreurs qui sautent le plus aux yeux à cause de leur absurdité. La première chose à noter est que ce comportement n’est pas causé par les descripteurs de couleur puisque leur fonctionnement dans ces circonstances reste prévisible. En effet, on peut s’attendre à ce que la moyenne et l’écart type aillent des valeurs hautes et que les histogrammes (R, G, B) soient très décalés vers la droite.

Pour comprendre le phénomène, il faut plutôt retourner consulter la figure No.14 qui illustre le comportement non prévisible du LBP lorsqu’il est appliqué sur une surface lisse comme de la neige. Puisqu’il s’agit d’une photo, les pixels dans un bloc de neige ont des valeurs très rapprochées mais non uniformes. Nous avons vu que le LBP est insensible aux variations d’intensité et qu’il traite les petites variations de la même manière que les grandes variations. Ainsi, les proportions des patrons binaires qui sont générés pour les blocs de neige sont complètement aléatoires et il en est de même pour les signatures générées. On peut donc conclure que ces erreurs de classification surviennent quand la signature LBP d’un bloc de neige est semblable à celle d’un tronc d’arbre.

Conclusion

Le système conçu est fonctionnel mais il faudrait lui apporter quelques améliorations pour qu’il soit plus fiable. En effet, comme discuté dans l’analyse il faudrait fournir plus d’images d’entrainement pour qu’il puisse reconnaître différents types de troncs. De plus, certaines de ces images devraient être prises lors d’autres saisons que l’hiver afin que le système puisse fonctionner toute l’année.

Une autre option à considérer est l’ajout de règles logiques pour détecter les classifications absurdes qui peuvent être faites par le système. Par exemple, quand un seul bloc de neige est classifié comme un tronc d’arbre. De plus, ces règles logiques pourraient remplir les trous qui surviennent quand la majorité des segments d’un tronc sont détectés mise à part quelques-uns.

Annexe

Figure No.1 : Exemples d'images à segmenter

Figure No.2 : Exemple de segmentation espérée

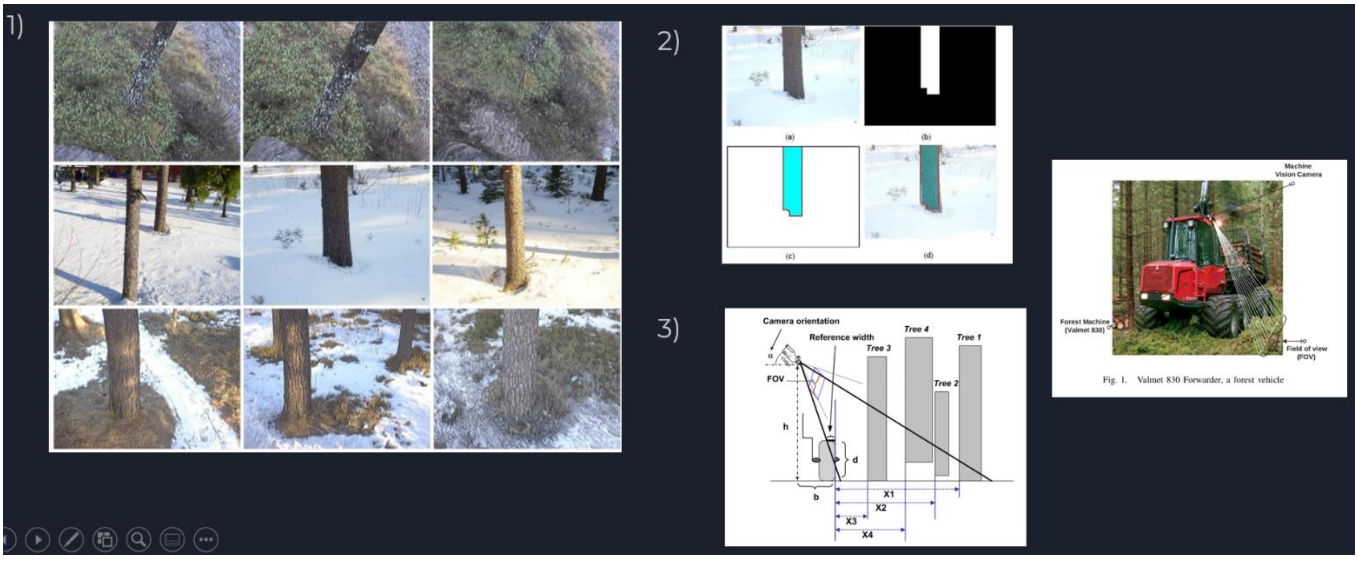


Figure No.3 : Projet de l'article de 2008 dans son ensemble

Figure No.4 : Exemples d'images à segmenter (dans l'article).

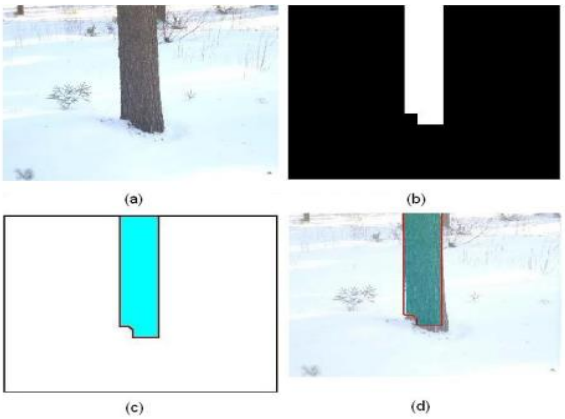


Figure No.5 : Exemple de segmentation (dans l'article)

```
avant normalisation         :  [40, 60, 10, 80, 10, 10, 10, 10, 10, 15, 35, 5, 80, 10, 15]
apres normalisation (/somme) :  [0.1, 0.15, 0.025, 0.2, 0.025, 0.025, 0.025, 0.025, 0.025, 0.0375, 0.0875, 0.01
25, 0.2, 0.025, 0.0375]
```

Figure No.7.1 : Allure d'un vecteur de données d'histogramme

```
avant normalisation        :  [128.4, 82.627]
apres normalisation (/255) :  [0.5035294117647059, 0.32402745098039215]
```

Figure No.7.2 : Allure d'un vecteur de données de statistiques

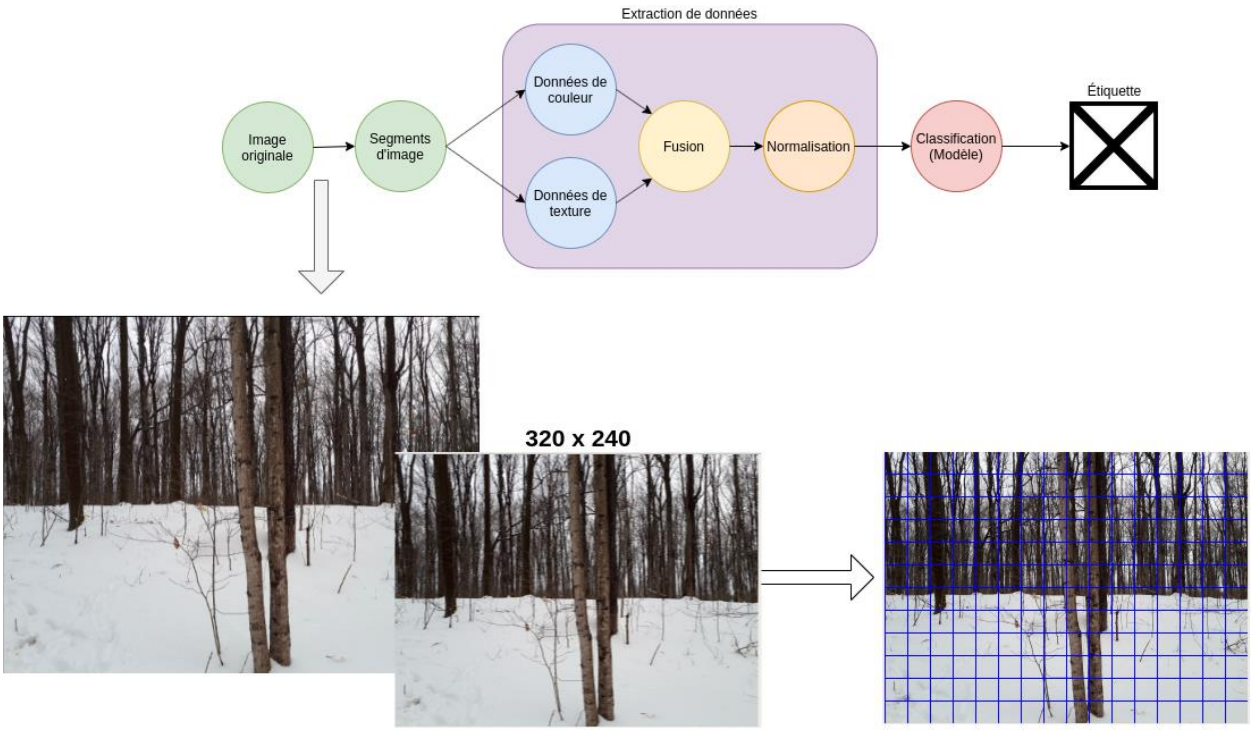
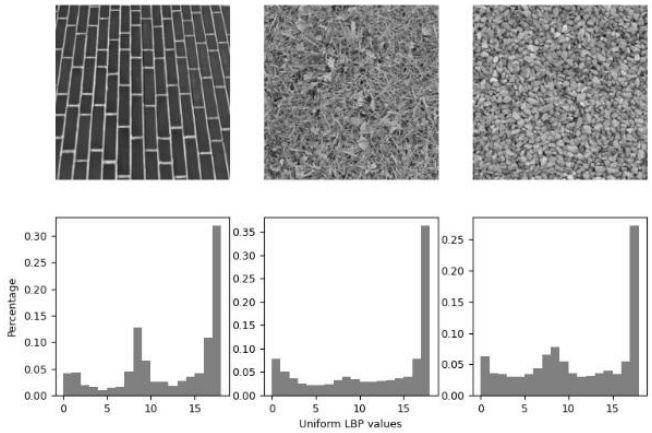


Figure No.6 : Schéma représentant l'approche générale



Les figures No.8 : signatures associées à différentes textures (exemple générique)

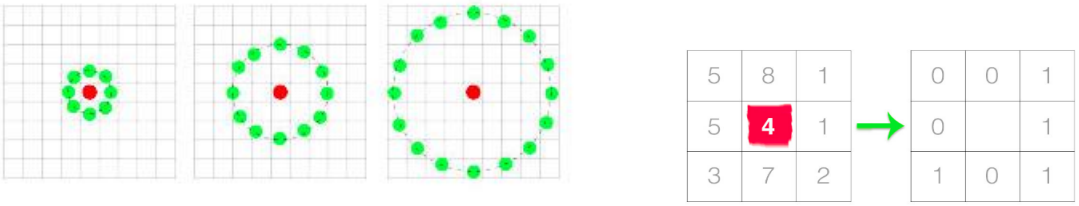
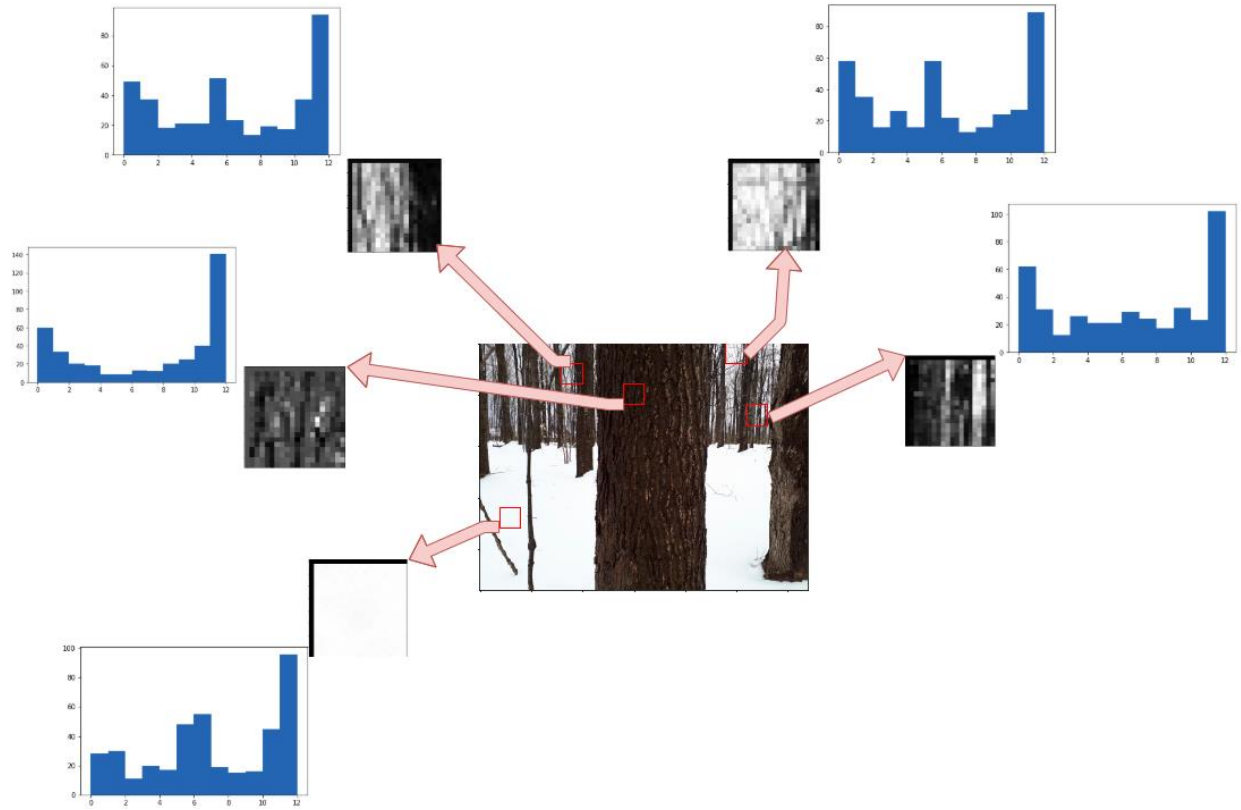
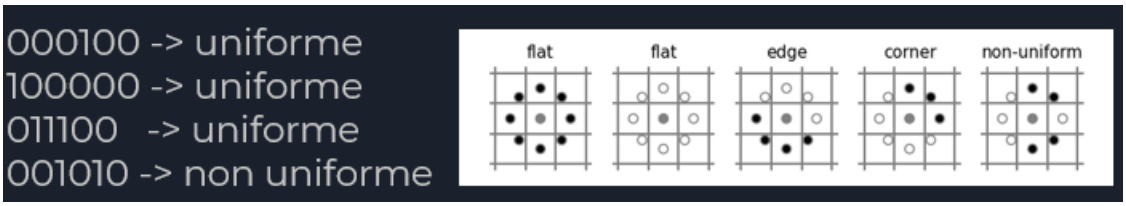


Figure No.10 : Processus de seuillage et cercles de voisinage.



Les figures No.9 : signatures associées à différentes textures (pour ce projet)



Figures No.11 : Exemples de patrons uniformes/non uniformes

```
avant normalisation          :  [40, 60, 10, 80, 10, 10, 10, 10, 10, 15, 35, 5, 80, 10, 15]
apres normalisation (/somme) :  [0.1, 0.15, 0.025, 0.2, 0.025, 0.025, 0.025, 0.025, 0.025, 0.0375, 0.0875, 0.01
25, 0.2, 0.025, 0.0375]
```

Figure No.13 : Allure d'un vecteur de données de texture

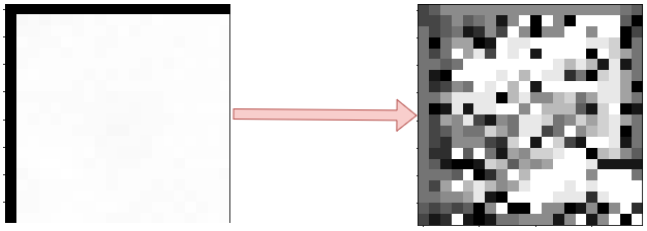
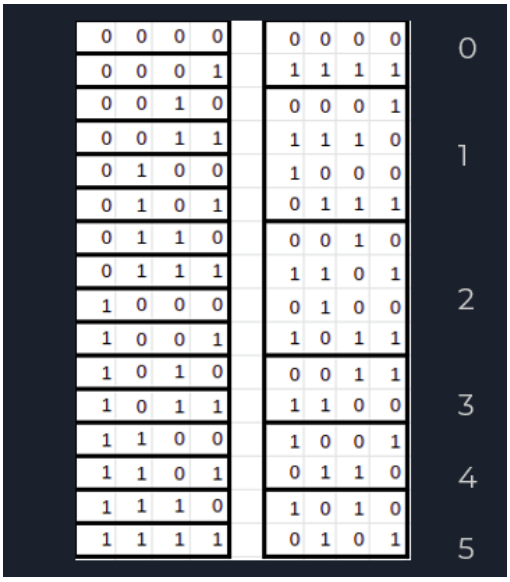


Figure No.14 : image LBP résultant pour un fond de neige



Figures No.12 : Exemple complet de classification de patrons pour un chaîne binaire de longueur 4

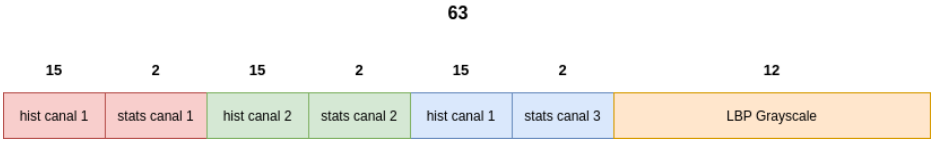


Figure No.15 : Méthode de fusion No.1

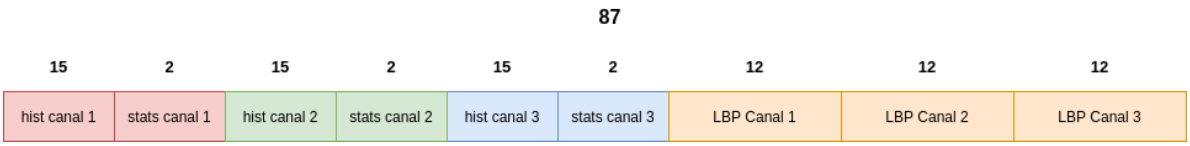


Figure No.16 : Méthode de fusion No.2

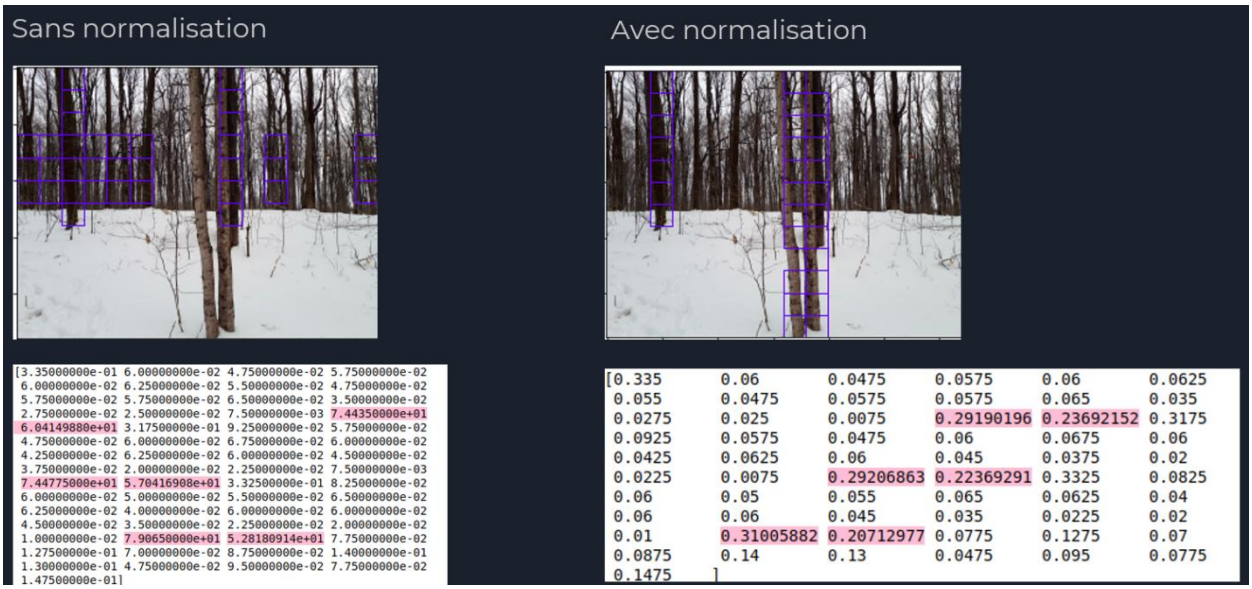


Figure No.17 : impact sur les résultats quand on retire les étapes de normalisation

$$précision = \frac{bonnes\ prédictions}{nombre\ total\ de\ prédictions} \qquad recall = \frac{true\ positives}{true\ positives\ + false\ negatives}$$

Figure No.18 : formule de précision classique et la formule de “recall”

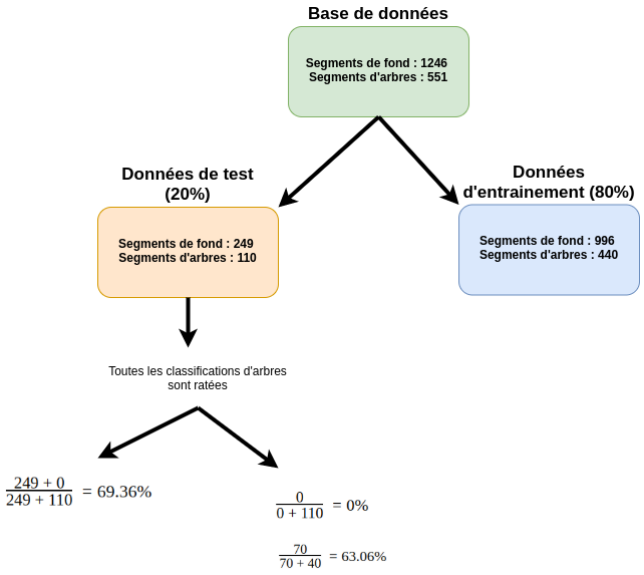


Figure No.19 : comportement des deux mesures de performance pour des données d’entrainement avec proportions inégales

```
"svm" : {
    "feature_extractor__channel_hist_n_bins" : [15, 20],
    "feature_extractor__lbp_n_points" : [8, 10, 15],
    "feature_extractor__lbp_radius" : [1, 2, 3],
    "feature_extractor__fusion_method" : [1, 2],
    "svm__kernel" : ["poly"],
    "svm__gamma" : ["scale"],
    "svm__C" : [0.1, 1, 100, 1000],
    "svm__degree" : [1, 2, 3]
}
```

Figure No.20 : grille de valeurs de paramètres utilisée pour la recherche par grille

```
{
    "input_img_size" : 20,
    "feature_extractor__color_space" : "RGB",
    "feature_extractor__channel_hist_n_bins" : 15,
    "feature_extractor__lbp_n_points" : 8,
    "feature_extractor__lbp_radius" : 1,
    "feature_extractor__fusion_method" : 1,
    "knn__n_neighbors" : 3,
    "svm__kernel" : "poly",
    "svm__C" : 100,
    "svm__degree" : 3
}
```

Figure No.21 : Les valeurs de paramètres retenues après la recherche

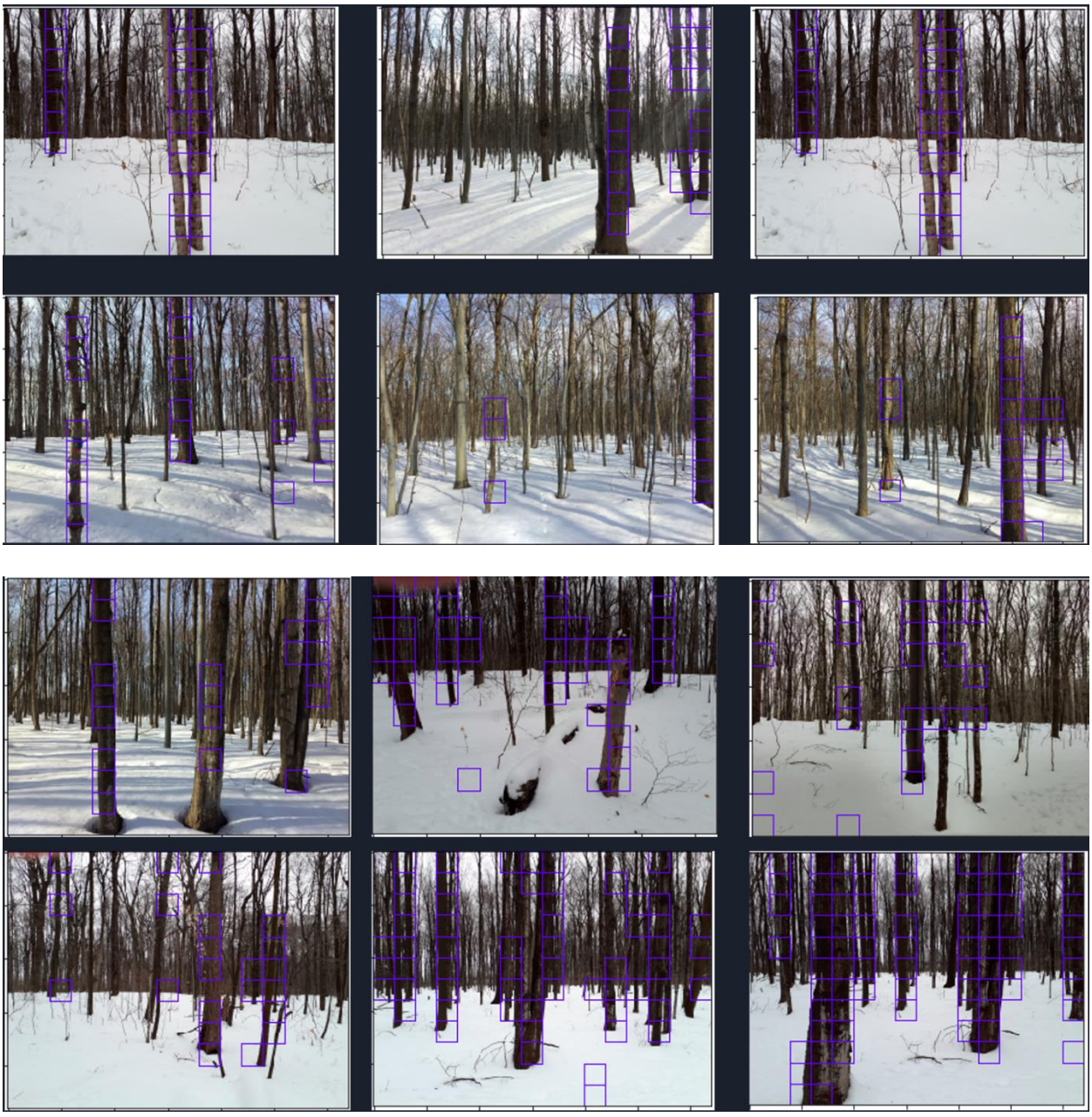


Figure No.22 : Performance du système face à des images qui ne sont pas dans l'ensemble d'entrainement