

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement

2017-2018

**Plateforme d'extraction et de
caractérisation automatique d'éléments
d'intérêt dans les images
archéologiques par apprentissage
interactif**

Tuteur académique

Jean-Yves RAMEL

Étudiant

Ronan GUILLAUME (DI5)

2 avril 2018



Liste des intervenants

Nom	Email	Qualité
Ronan GUILLAUME	ronan.guillaume@etu.univ-tours.fr	Étudiant DI5
Jean-Yves RAMEL	jean-yves.ramel@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Ronan GUILLAUME susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Jean-Yves Ramel susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Ronan GUILLAUME, *Plateforme d'extraction et de caractérisation automatique d'éléments d'intérêt dans les images archéologiques par apprentissage interactif*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2017-2018.

```
@mastersthesis{
  author={GUILLAUME, Ronan},
  title={Plateforme d'extraction et de caractérisation automatique d'éléments d'intérêt
    dans les images archéologiques par apprentissage interactif},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2017-2018}
}
```


Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	v
1 Introduction	1
1 Les acteurs du projet	1
2 Contexte	2
3 Description du problème	2
4 Objectifs	3
5 Bases méthodologiques	3
I Recherche	4
2 Description générale	5
1 Environnement du projet	5
2 Caractéristiques des utilisateurs	5
3 Sources fournies par la MOA	5
4 Fonctionnalités et structure générale du système	6
4.1 Processus de classification.....	6
4.2 Diagramme de cas d'utilisation.....	6
5 Existant	7

3	État de l'art	8
1	Intelligence artificielle.....	8
2	Reconnaissance de formes.....	8
2.1	Acquisition des données.....	9
2.2	Pré-traitement.....	9
2.3	Analyse.....	9
2.4	Apprentissage	9
2.5	Décision.....	10
3	Type d'apprentissage automatique	10
3.1	Apprentissage supervisé.....	10
3.2	Apprentissage non-supervisé	10
3.3	Apprentissage par renforcement.....	11
4	Classifieur linéaire	11
5	Séparateurs à Vaste Marge (SVM)	11
6	Réseaux de neurones à convolutions (CNN).....	12
6.1	Perceptron, le neurone des ordinateurs	12
6.2	Neurone sigmoïde.....	13
6.3	Principe	13
6.4	Couche de convolution (CONV).....	14
6.5	Couche de pooling (POOL)	14
6.6	Couche de correction (ReLU).....	14
6.7	Couche de "entièrement connectée" (FC)	15
6.8	Couche de perte (LOSS).....	15
6.9	Exemple de modèle[6]	15
4	Analyse et conception	17
1	Librairies utilisées	17
2	Diagramme de classes	17
II	Développement	18
5	Découpage du projet en tâches	19
1	Sprint 1 : Prise en main du projet	20
1.1	Description de la tâche	20
1.2	Livrable	20
1.3	Estimation de charge	20
2	Sprint 2 : Workspace.....	20
2.1	Description de la tâche	20
2.2	Livrable	20

2.3	Estimation de charge	20
3	Sprint 3 : Sélection des caractéristiques à exploiter	21
3.1	Description de la tâche	21
3.2	Livrable	21
3.3	Estimation de charge	21
4	Sprint 4 : Modification des images de résultats.....	21
4.1	Description de la tâche	21
4.2	Livrable	22
4.3	Estimation de charge	22
5	Sprint 5 : Mise en place de l'IHM.....	22
5.1	Description de la tâche	22
5.2	Livrable	23
5.3	Estimation de charge	23
6	Sprint 6 : Rédaction du rapport	23
6.1	Description de la tâche	23
6.2	Livrable	23
6.3	Estimation de charge	23
6	Implémentation	24
1	Analyse de l'existant.....	24
1.1	Variables statiques	24
1.2	Convention de nommage.....	25
1.3	Propriétés en Python	25
2	Workspace.....	26
2.1	Evolutivité de l'arborescence requise.....	26
2.2	Parcours de l'arborescence.....	27
3	Sélection des caractéristiques à exploiter	28
3.1	Constante all_definition_features.....	28
3.2	Conception de la classe Feature	28
3.3	Evolutivité des features	31
4	Modification des images de résultats.....	31
4.1	Nouvelle conception	31
4.2	Modification de l'image comparative.....	32
5	Mise en place de l'IHM.....	33
5.1	Élaboration de la MockUp.....	33
5.2	Implémentation des vues	33
5.3	Vues créées.....	33

7	Qualité mise en œuvre	36
1	Documentation	36
2	Tests unitaires	36
3	Tests de validation	37
8	Récapitulatif	38
9	Conclusion	39
	Annexes	40
A	Diagramme de cas d'utilisation	41
B	Cahier des spécifications	42
	Webographie	59
	Bibliographie	60

Table des figures

1 Introduction

1	Logo du groupe RFAI du Laboratoire Informatique de Tours.....	1
2	Logo du projet SOLiDAR et du laboratoire CITERES de Tours	2
3	Exemple d'image de relief prise à partir d'image LiDAR.....	2
4	Comparaison vue aérienne et acquisition par télédétection LiDAR	3
5	Reconnaissance des différents reliefs.....	3

2 Description générale

1	Exemple des images sources à fournir par la MOA	6
2	Processus de classification[1].....	6

3 État de l'art

1	Récapitulatif des différentes étapes d'un système de reconnaissance de formes	9
2	Schéma de classification de formes suite à un apprentissage[2].....	10
3	Classifieur à trois classes utilisant deux caractéristiques[1]	11
4	Exemple de problème linéairement et non-linéairement séparable[5].....	11
5	Exemple d'une frontière de classe de svm optimisant la marge séparant deux classes[1]	12
6	Schéma simplifié d'un neurone[3].....	12
7	Schéma d'un perceptron[3]	13
8	Exemple d'un perceptron[3]	13
9	Exemple du processus de convolution[2]	14
10	Exemple du processus de pooling Max-Pool 2x2[6]	15
11	Architecture standard d'un réseau à convolutions.....	15

5 Découpage du projet en tâches

1	Diagramme de Gantt	19
2	Récapitulatif des fonctionnalités du workspace	21
3	Image comparative actuelle.....	22

6 Implémentation

1	Classe Workspace	26
2	Classe ImageFeaturesExtractor	28
3	Conception des classes de features de l'existant.....	28
4	Exemple de local binary patterns.....	31
5	Nouvelle conception des résultats	32
6	Système de gestion des couleurs HSV.....	32
7	Nouvelle version des résultats.....	32
8	QtDesigner, outil de création d'interfaces graphiques	33
9	Vue de démarrage	34
10	Vue principale reprenant le nom des classes et d'images d'entraînement.....	34
11	Tab permettant d'extraire les features	34
12	Tab permettant d'interagir avec le modèle de reconnaissance	34
13	Vue montrant les résultats d'une évaluation	35

8 Récapitulatif

1	Récapitulatif du projet	38
---	-------------------------------	----

A Diagramme de cas d'utilisation

1	Diagramme de cas d'utilisation	41
---	--------------------------------------	----

1

Introduction

Dans le cadre de la formation en ingénieur informatique à l'école Polytechnique de Tours, un Projet de Recherche & Développement (PR&D) est réalisé en 5ème année afin de constituer une réelle expérience en terme de conduite de projet. Cette année, le PR&D se déroule du 11 septembre 2017 au 5 avril 2018 à raison de 2 jours par semaine à temps plein. Il donne lieu à la rédaction d'un rapport et de deux présentations du travail effectué lors de deux soutenances.

Le PR&D est orienté vers une conception logiciel de reconnaissance de formes où la MOA est représentée par Clément LAPLAIGE et Xavier RODIER, chercheurs en archéologie du laboratoire LAT - CITERES et Mr Jean-Yves RAMEL, enseignant chercheur à l'école Polytechnique de Tours. Le projet s'inscrit dans un cadre théorique qui fait l'objet d'une étude algorithmique en amont, pour aboutir à l'amélioration d'un logiciel.

Ce document est donc le rapport de recherche du projet « Plateforme d'extraction et de caractérisation automatique d'éléments d'intérêt dans les images archéologiques par apprentissage interactif ». Il définit l'état de l'art, l'environnement du projet, les objectifs à réaliser, les méthodes de mises en œuvre ainsi que les résultats.

1 Les acteurs du projet

- La maîtrise d'œuvre (MOE) : Ronan GUILLAUME, étudiant en 5ème année de l'Ecole d'Ingénieur Polytech Tours au sein du département Informatique, dans le cadre du PR&D ainsi que son encadrant Jean-Yves RAMEL de l'équipe de recherche RFAI - EA 6300 du Laboratoire Informatique de Tours (<http://www.rfai.li.univ-tours.fr>).



Figure 1 – Logo du groupe RFAI du Laboratoire Informatique de Tours

Les recherches menées au sein de cette équipe portent sur l'étude et la résolution des problèmes de reconnaissance de formes, que ce soit dans le cadre de problématiques "industrielles" ou de problèmes théoriques.

- La maîtrise d'ouvrage (MOA) : Le projet SOLiDAR du laboratoire Archéologique et Territoire CITERES de Tours, représenté par Clément LAPLAIGE et Xavier RODIER. Créé en 2004, ce laboratoire basé sur Tours a pour objectif d'analyser les dynamiques spatiales et territoriales des sociétés. Dans ce contexte, leur champ de recherche se compose de quatre secteurs : la recherche urbaine, la recherche environnementale, les travaux sur le territoire et ceux sur les effets des recompositions sociales contemporaines. (<http://citeres.univ-tours.fr>)



Figure 2 – Logo du projet SOLiDAR et du laboratoire CITERES de Tours

Ce document a été rédigé par Ronan GUILLAUME et fera l'objet d'une note en fin de semestre.

2 Contexte

Les archéologues travaillent avec un programme de télédétection basé sur une technologie dite LiDAR (« Light Detection And Ranging ») qui permet d'acquérir des images géographiques. A partir de ces acquisitions, les archéologues génèrent plusieurs types d'images selon leurs besoins et l'un d'entre eux est l'image de relief sur laquelle les archéologues peuvent marquer manuellement les éléments archéologiques.

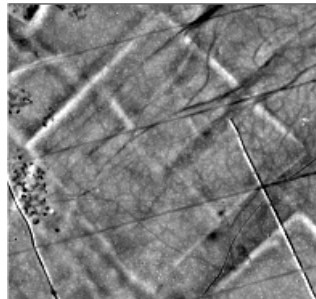


Figure 3 – Exemple d'image de relief prise à partir d'image LiDAR

Les archéologues souhaitent réaliser un logiciel qui permettrait d'automatiser la détection d'éléments archéologiques dans des images de relief. Cette solution doit permettre de générer un tableau de probabilité et non une décision binaire, de plus, il doit être open-source et doit permettre à l'utilisateur d'exploiter facilement sa propre banque de données.

C'est donc dans ce contexte que la MOA propose un projet afin de trouver une solution pour prédire les probabilités pour chaque pixel de l'image son appartenance à chaque classe de forme.

3 Description du problème

A l'aide la technologie LiDAR, il est possible de détecter des micro-reliefs sous couvert forestier révélant des structures archéologiques ou naturelles qui ne sont pas toujours visible à l'œil nu.

Les archéologues doivent ensuite manuellement les relever, les analyser puis les caractériser suivant différents types de forme. Le problème consiste à automatiser ce processus de reconnaissance.

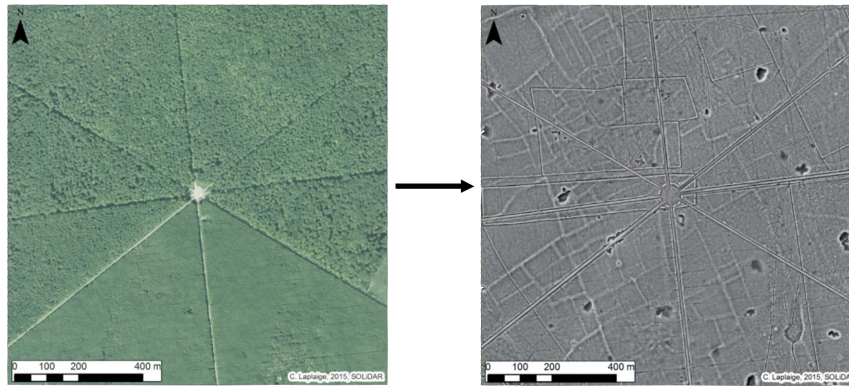


Figure 4 – Comparaison vue aérienne et acquisition par télédétection LiDAR

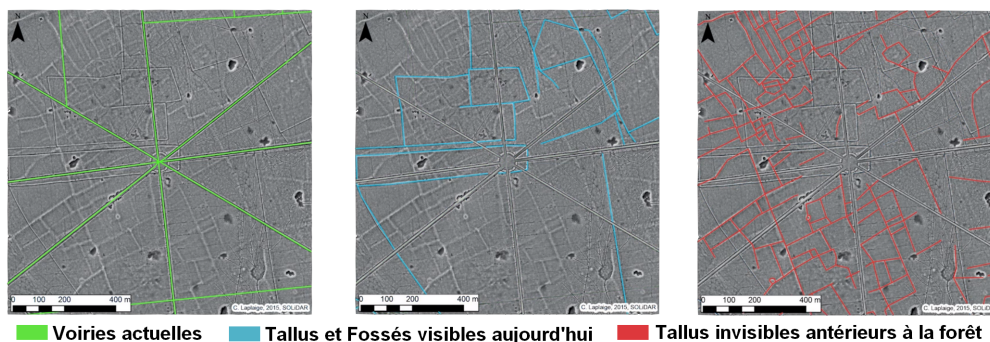


Figure 5 – Reconnaissance des différents reliefs

4 Objectifs

Ce projet a pour objectif de reprendre un existant de l'année passé et de poursuivre son développement. Cela passe par ces points suivants :

- Revoir l'interface homme-machine afin de la rendre plus ergonomique.
- Revoir la conception du projet afin d'assurer la généricité du projet.
- Générer de la documentation interne et externe au code afin de faciliter la reprise du code pour un développeur externe au projet.
- Améliorer les performances en termes de taux de reconnaissance.
- L'ajout de fonctionnalités permettant à l'utilisateur de sélectionner des caractéristiques.
- L'ajout de fonctionnalités permettant à l'utilisateur de sélectionner/paramétrer son classifieur.

5 Bases méthodologiques

En terme d'organisation de développement, le projet s'appuiera sur la méthodologie agile SCRUM. Basé sur le découpage du projet en plus petites tâches, celle-ci seront remplies lors de courtes périodes nommées "sprints" dont l'unité de mesure sera en semaine ici. Chaque sprint se terminera par une livraison du produit ainsi que d'une rétrospection afin d'assurer une amélioration du processus de réalisation tout au long du projet.

Première partie

Recherche

2

Description générale

1 Environnement du projet

Ce projet sera réalisé en Python3.5 à l'aide de l'IDE PyCharm, logiciel de la distribution JetBrains et gratuit pour les étudiants.

Un Git est utilisé afin de gérer le versionning du projet ainsi qu'un drive Google afin de partager certains documents avec la MOA.

2 Caractéristiques des utilisateurs

Le produit final a pour but d'être utilisé par des archéologues. Nous devons donc prendre en compte les requis suivants pour que les futurs utilisateurs puissent se servir de cette solution :

- Connaissances basiques de l'informatique;
- Utilisateurs réguliers;
- Connaître le fonctionnement de ou des algorithmes de résolution afin de pouvoir les paramétrer.

Ce dernier point permet de mettre en exergue la possibilité d'avoir deux types d'utilisateurs, un ayant les connaissances requises pour modifier les paramètres de résolutions et un utilisateur régulier ne se contentant qu'à utiliser les fonctionnalités principales de la solution sans se soucier de son paramétrage.

Il sera donc primordial que l'IHM permette à ce deuxième type d'utilisateurs d'être guidé tout au long de son utilisation.

3 Sources fournies par la MOA

La MOA doit fournir deux types d'images sources : des images dites "originelles" en niveau de gris ainsi que des images dites "labellisées" binaires. Ces deux types d'images seront au format TIFF (256 nuances, 8 bits).

Les images labellisées représentent une image originelle pour une classe de reconnaissance (elles devront avoir le même nom). Un pixel noir dans cette image binaire signifie que ce pixel appartient à la classe que concerne l'image labellisé.

Dans l'exemple ci-dessous, les images sources permettent de différencier cinq classes différentes dans ce modèle. Nous pouvons remarquer ici que seuls les classes trois et quatre sont présentes dans cette image source. Le nom en-dessous de chaque image correspond au nom du dossier dans lequel elles se trouvent respectivement. En effet, si elles doivent toutes avoir le même nom, elles ne peuvent donc pas être stockées dans le même répertoire.

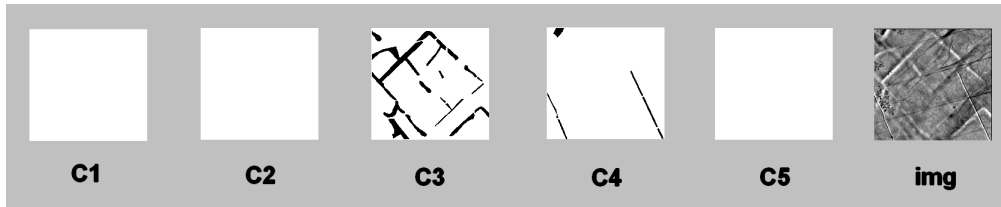


Figure 1 – Exemple des images sources à fournir par la MOA

4 Fonctionnalités et structure générale du système

4.1 Processus de classification

Ce système a pour but "d'apprendre", à partir d'images sources, à classer les pixels d'une image suivant différentes formes.

Une fois l'apprentissage effectué, le modèle aura généré un classifieur lui permettant d'attribuer à chaque pixel d'une image, sa probabilité d'appartenance pour chaque forme.

En fonction de son choix de méthode d'apprentissage, il sera nécessaire à l'utilisateur de définir les caractéristiques que le modèle pourra exploiter. De plus, il pourra paramétrer le classifieur.

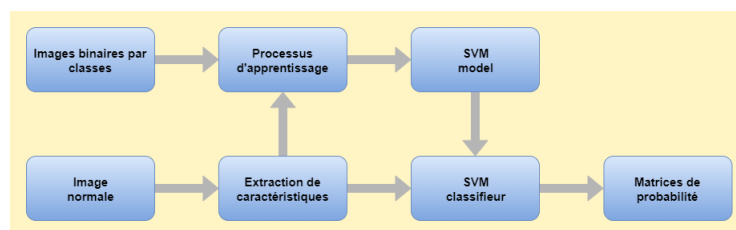


Figure 2 – Processus de classification[1]

4.2 Diagramme de cas d'utilisation

Les différents cas d'utilisation du système sont explicités dans le diagramme en annexe A. La fonctionnalité principale de l'outil est bien sûr de pouvoir classer une image afin de visualiser les différentes formes que le modèle a reconnu. Cependant, le modèle doit avoir appris au préalable à différencier les différentes classes de formes. L'utilisateur doit donc pouvoir lancer un apprentissage ainsi que de tester son modèle pour vérifier que ce dernier est suffisamment compétent pour reconnaître efficacement ces formes. Enfin, l'utilisateur doit pouvoir configurer son classificateur s'il souhaite ajouter une caractéristique, ou d'une manière plus générale paramétrer son classifieur.

5 Existant

Ce projet se base grandement sur les travaux effectués lors d'années précédentes. Le diagramme de cas d'utilisation en Annexe A montre en jaune les fonctionnalités déjà implémentées et en orange celle qui le sont partiellement.

Même si l'existant est nettement améliorable, il est déjà fonctionnel.

3

État de l'art

Ce chapitre définira les concepts de l'intelligence artificielle, de reconnaissance de formes, et plus particulièrement ses méthodes d'apprentissage.

1 Intelligence artificielle

Ce terme pouvant être abrégé par le sigle IA fut défini comme étant "la construction de programmes informatiques qui s'adonnent à des tâches qui sont, pour l'instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique"[4] par l'un des créateurs de ce concept, Marvin Lee Minsky en 1956.

L'une des premières recherches effectuée dans cette matière fut un article d'Alan Turing publié en 1950 dans lequel il propose un test afin de pouvoir qualifier ou non une machine de "consciente" connue désormais sous le nom de test de Turing. Ce test est le suivant : un juge dialogue avec un humain et une machine à l'aide d'un terminal. Si le juge n'arrive pas à différencier l'humain de la machine, alors le test est validé.

De nos jours, aucune machine n'a réussi à passer ce test. Mais avec les avancées technologiques en la matière, il est possible de s'imaginer que l'on y arrive un jour. Effectivement, même si nos machines ne sont pas encore capables de penser, elles sont déjà capable d'apprendre. Par exemple, le programme AlphaGo de l'entreprise Google a battu au jeu de go le champion du monde en mars 2016. Ce jeu était le dernier jeu où l'homme était meilleur que la machine.

2 Reconnaissance de formes

L'une des applications de l'intelligence artificielle, et dont relève ce projet, est la reconnaissance des formes. Il s'agit d'un domaine regroupant toutes les méthodes permettant d'identifier des motifs informatiquement. Ses domaines d'applications sont très variés : Reconnaissance de paroles, de locuteur, de textes, en radiologies, pour l'identification d'objets, de défauts, pour la prévision météorologique, boursière...

Dans la très grande majorité des cas, la reconnaissance de formes s'appuie sur un apprentissage automatique afin de créer un modèle lui permettant de classifier différents motifs. Ensuite, ce modèle est exploité afin de reconnaître les différents motifs dans une image source.

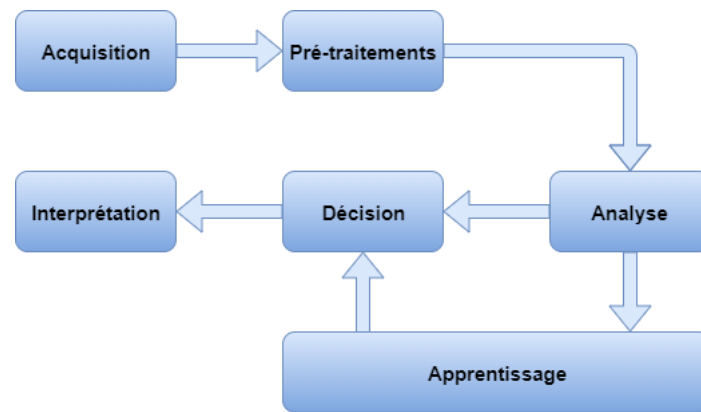


Figure 1 – Récapitulatif des différentes étapes d'un système de reconnaissance de formes

2.1 Acquisition des données

Cette première étape correspond à l'acquisition de données numériques exploitables par le modèle.

2.2 Pré-traitement

Le pré-traitement correspond aux retouches que l'on pourrait apporter aux données sources. Cela peut correspondre à l'ajout ou l'élimination de bruit, la suppression de redondances, un lissage, un réhaussement des contrastes, etc

2.3 Analyse

L'analyse correspond à l'extraction de caractéristiques ou d'indices de données.

Si un modèle souhaite différencier plusieurs types de véhicules : les motos, les voitures et les bus. Une caractéristique que le modèle pourrait extraire serait le nombre de roues présentes sur chacune des images sources, ou le rapport entre la hauteur et la longueur du véhicule. L'intérêt de ces caractéristiques est qu'elles vont pouvoir être exploitées pour reconnaître la forme. En effet, il est évidemment qu'un véhicule possédant quatre roues et une longueur trois fois plus grande que sa hauteur est bien un bus.

Une caractéristique est donc une mesure, ou un ensemble de mesures permettant de décrire la forme présente sur l'image source, cela peut correspondre à la moyenne des valeurs rouge, verte et bleue d'un voisinage de chaque pixel d'une image ou bien la différence de valeur d'un pixel par rapport à son voisinage.

Il faut remarquer que l'efficacité du modèle dépendra du choix de ces caractéristiques. En effet, s'il est facile de deviner un véhicule à partir du nombre de roues et de son rapport hauteur/longueur, il est beaucoup plus difficile si est seulement extrait la couleur du véhicule.

2.4 Apprentissage

L'apprentissage correspond à la partie où le logiciel va "apprendre" à différencier les différentes formes. Pour cela, un algorithme d'apprentissage automatique sur les caractéristiques d'une base de sources déjà classifiées (Pour plus de détail sur les apprentissages automatiques, voir page 10).

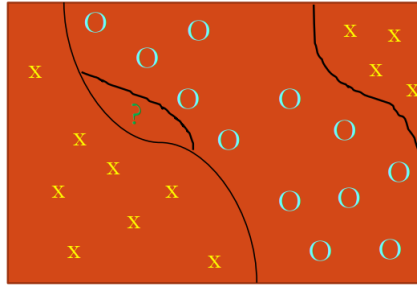


Figure 2 – Schéma de classification de formes suite à un apprentissage[2]

Dans l'exemple ci-dessus, il est possible d'observer une visualisation à deux dimensions du résultat d'un apprentissage de deux formes X et O. Chaque lettre correspondant à une image source, le modèle a défini plusieurs zones d'appartenance pour chaque forme. Il est notable qu'il peut y avoir des zones d'indétermination (représenté par un "?") si les des éléments des deux formes ont été relevés dans cette zone ou aucune des deux.

2.5 Décision

Une fois l'apprentissage effectué, le modèle peut alors reconnaître des formes. Par analogie à l'exemple ci-dessus, Il ne suffit au programme qu'à observer dans quel zone se situe l'image qui lui est passée en entrée. Il pourra alors définir si il s'agit d'un X ou d'un O et dans le cas contraire, que sa forme est indéterminée.

3 Type d'apprentissage automatique

Comme il a été vu précédemment (Voir 2.4), l'apprentissage automatique désigne l'ensemble des méthode permettant à un ordinateur à "apprendre" afin de pouvoir faire des prédictions. Les algorithmes d'apprentissage varient en fonction du mode d'apprentissage qu'ils emploient.

3.1 Apprentissage supervisé

Lorsque le nombre de classes est connu et que chaque image de la base d'apprentissage a été étiquetée au préalable, on parle d'apprentissage supervisé. Le nombre de classes étant fixe, il est possible pour le logiciel de calculer une probabilité d'appartenance pour chaque classe.

Parmi les algorithmes d'apprentissage supervisé peuvent être cités la Machine à Vecteurs de Support (SVM), le réseau de neurones, l'Analyse Discriminante Linéaire (LDA)...

3.2 Apprentissage non-supervisé

A contrario, si le système n'a pas connaissance des classes à reconnaître, on parle d'apprentissage non-supervisé. L'intérêt de cette méthode est qu'il n'y a plus besoin d'expert pour caractériser chaque image d'entraînement. C'est donc au logiciel de créer des "communautés" entre ses différentes images sources qu'il considérera comme étant les classes qu'il aura reconnu.

3.3 Apprentissage par renforcement

L'apprentissage par renforcement se base sur le concept de récompense. Le principe de cet apprentissage est l'itération d'un agent autonome qui va prendre des décisions. Pour chaque bonne décisions, l'agent sera récompensé, et inversement. Il cherchera donc à maximiser ses récompenses.

4 Classifieur linéaire

Le classifieur linéaire est l'entité permettant de regrouper les échantillons en classes à partir de caractéristiques similaires.

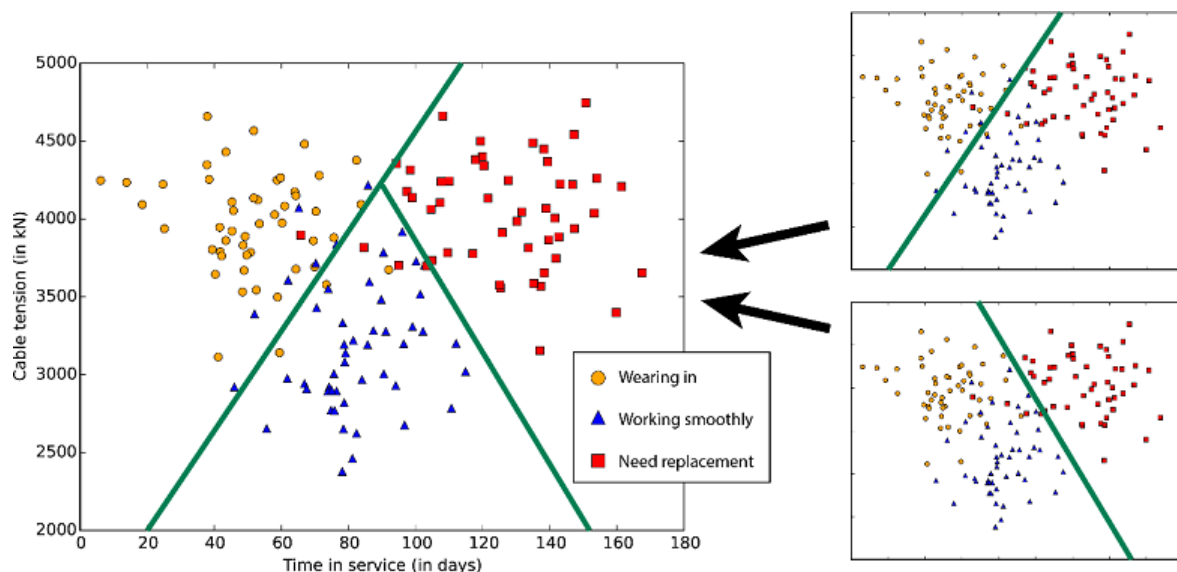


Figure 3 – Classifieur à trois classes utilisant deux caractéristiques[1]

Le classifieur étant linéaire, seuls les problèmes linéairement séparables pourront être traité.

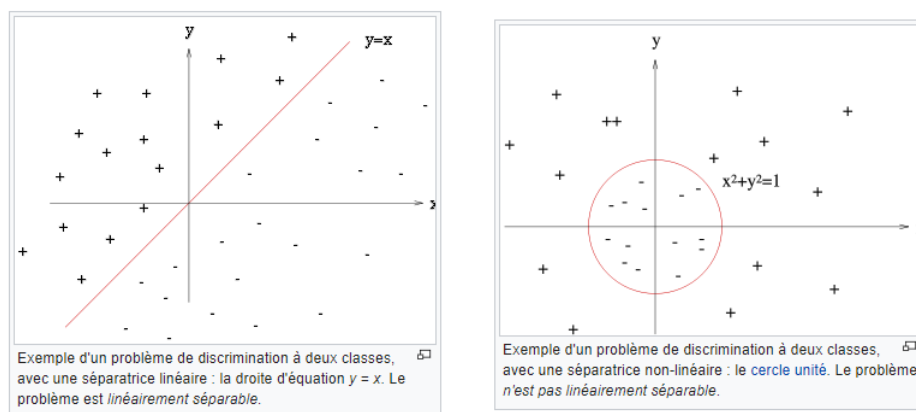


Figure 4 – Exemple de problème linéairement et non-linéairement séparable[5]

5 Séparateurs à Vaste Marge (SVM)

Les séparateurs à vaste marge sont un classifieur se basant sur l'idée de marge maximale. La marge étant la distance entre la frontière de séparation et les échantillons les plus proches. Cette

recherche maximale de la marge assure la qualité du modèle en s'appuyant sur la théorie de Vapnik-Chervonenkis (ou théorie statistique de l'apprentissage).

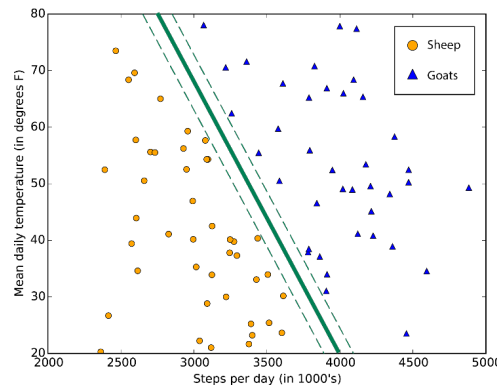


Figure 5 – Exemple d'une frontière de classe de svm optimisant la marge séparant deux classes[1]

Ce classifieur a pour avantage de pouvoir séparer linéairement des données même si elles ne l'étaient pas à la base grâce une fonction noyau plutôt coûteuse en performance. Le principe de cette astuce est de reconsidérer le problème dans un espace de dimension plus grand voir infini. Il est alors probable qu'il existe une séparation linéaire dans ce nouvel espace.

6 Réseaux de neurones à convolutions (CNN)

Le réseaux de neurones à convolutions est un classifieur dit bio-inspiré puisque qu'il imite le cortex visuel des animaux. Il a l'avantage de ne pas avoir de pré-traitements mais nécessite une base d'apprentissage plus conséquente que le SVM.

6.1 Perceptron, le neurone des ordinateurs

Tout d'abord, d'un point de vu biologique, le neurone est une cellule permettant de transférer un signal bioélectrique. Il est constitué d'un noyau, d'un long axone lui permettant d'envoyer le potentiel d'action et d'un très grand nombre de dendrites lui permettant de recevoir les potentiels d'autres neurones.

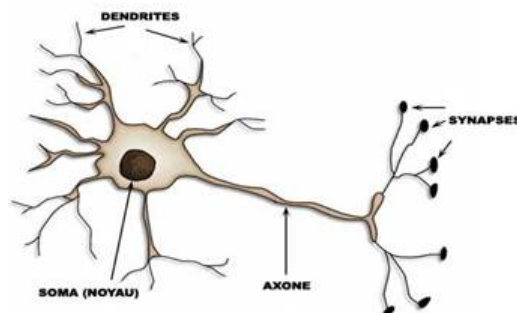


Figure 6 – Schéma simplifié d'un neurone[3]

De manière similaire, les neurones artificiels (appelés perceptrons) sont caractérisés par un certains nombres d'entrées binaires et d'une sortie binaire également.

L'utilisation d'un exemple permettra de comprendre le fonctionnement d'un perceptron. Tout d'abord, pour chaque entrée, un perceptron possède un poids ω_i . De plus, chaque perceptron

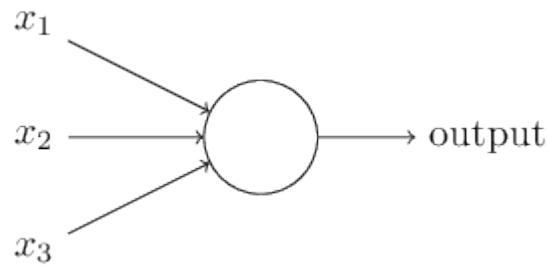


Figure 7 – Schéma d'un perceptron[3]

possède un seuil fixe θ . Le potentiel d'action d'un perceptron est alors calculé de la manière suivante :

$$z = \sum_i (\omega_i * x_i) + \theta$$

Dans cet exemple, le potentiel de ce perceptron pour (0;0) est 3 tandis que pour (1;1) il vaut -1 (Pour rappel, x_i est binaire).

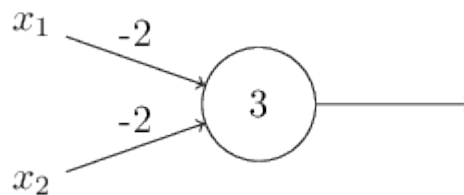


Figure 8 – Exemple d'un perceptron[3]

Le perceptron va ensuite s'activer si son potentiel est strictement positif.

Ce type de perceptron est limité par la binarité de ses entrées et sortie. En effet, le moindre petit changement de poids ou de son seuil peut causer des résultats complètement différents, ce qui est un problème pour l'apprentissage.

6.2 Neurone sigmoïde

L'utilisation de neurones sigmoïdes permet de pallier au problème ci-dessus. À la différence du perceptron, il reçoit en entrée et renvoie en sortie un nombre entre 0 et 1.

En effet, la sortie est définie par le résultat de la fonction sigmoïde :

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

6.3 Principe

Le principe de cette méthode d'apprentissage est d'utiliser les neurones sigmoïdes afin d'analyser l'image bloc par bloc, aussi appelé tuile. Ce choix d'implémentation est motivé par le choix de diminuer grandement le nombre de connections entre chaque perceptron.

Pour ce faire, l'architecture modulable du réseau de neurones convolutifs se constitue d'une succession de différentes couches de traitement décrites ci-dessous.

Lors de l'apprentissage, une rétroaction sera effectuée après chaque image d'entraînement afin d'ajuster les poids ω_i ainsi que les seuils θ de chaque perceptron.

6.4 Couche de convolution (CONV)

La couche convolution permet de retrouver des motifs à l'intérieur d'une image en les comparant à un champs récepteur. Cela permet de faire ressortir des ressemblance de motif qui serait juste légèrement décalé dans l'image.

D'un point de vue mathématiques, cela est très simple. Pour calculer la correspondance entre un champs récepteur et une tuile de notre image, il suffit de multiplier chaque pixel du champ récepteur avec son homologue sur l'image. Ensuite, il faut sommer ces multiplications et diviser le résultat par le nombre total de pixels du champs de réception. Si les 2 pixels sont de couleur identique, le résultat de leur multiplication vaudra 1, -1 dans le cas contraire.

Si tous les pixels dans un champs de réception correspondent avec une portion de l'image, alors leur addition puis leur division par le nombre total de pixels donne 1. De la même manière, si aucun des pixels ne correspond à la sous-partie de l'image, alors la réponse est -1.

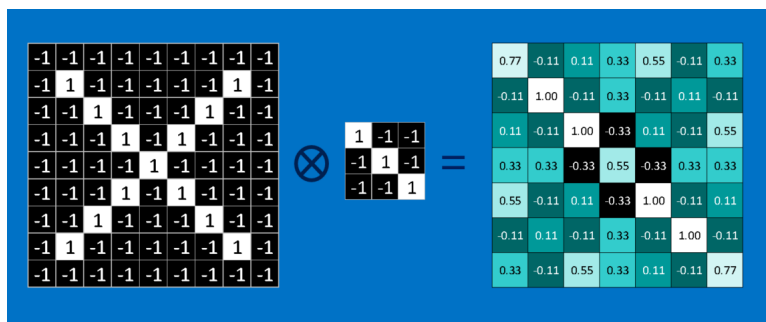


Figure 9 – Exemple du processus de convolution[2]

Plusieurs paramètres sont modifiables dans ce type de couche :

- Profondeur de la couche : c'est-à-dire le nombre de perceptron associé au champ récepteur (autrement dit, la taille de ce dernier).
- Le pas contrôle le chevauchement des champs récepteurs. Dans l'exemple ci-dessus le pas est de 1 mais il pourrait être augmenté afin de diminuer le volume de sortie.
- La marge (à 0) ou zero padding : il est parfois utile de mettre des zéros à la frontière du volume d'entrée. Cette "marge" permet de prendre en compte la dimension spatiale de la sortie. En effet, il peut-être souhaitable de conserver la même surface que celle du volume d'entrée.

6.5 Couche de pooling (POOL)

Le pooling est une couche utilisé entre deux couches de convolution afin de réduire le sur-apprentissage. Il peut-être défini comme étant une compression de l'image d'entrée. L'image est découpé en carré de taille N pixels et l'image de sortie est donc N fois plus petite. La valeur de cette nouvelle tuile dépend de la fonction de pooling mais la plus utilisé est le "Max-Pool 2x2" récupérant la valeur max de chaque carré de taille 2 pixels.

L'utilisation de pooling repose sur un compromis entre la perte d'information, puisque l'image est grandement compressé, et la puissance de calcul.

6.6 Couche de correction (ReLU)

Cette couche s'intercale entre des couches de traitement afin de rectifier certaines données qui pourrait compromettre le bonne apprentissage du modèle. L'exemple le plus utilisé est la

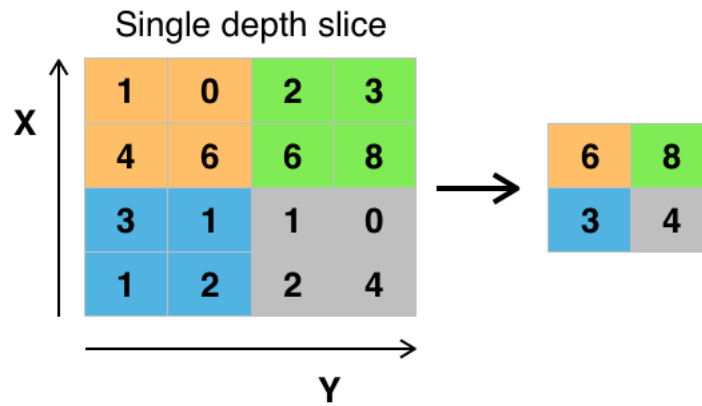


Figure 10 – Exemple du processus de pooling Max-Pool 2x2[6]

fonction d'activation non saturante permettant de remplacer les valeurs négatives par 0.

6.7 Couche de "entièrement connectée" (FC)

Après un certain nombre d'itérations entre convolution et pooling, il est possible de faire appel à une couche entièrement connectée. Elle considère l'image entièrement et son nombre de perceptron est donc nettement supérieur. Cependant, ce nombre fut grandement diminué par les couches de pooling. De plus, chaque perceptron de la couche n-1 sera connecté à tous les perceptrons de la couche n.

6.8 Couche de perte (LOSS)

Enfin, la couche de perte correspond à la rétrospection lors de l'entraînement du modèle. Il pénalise la différence entre l'attendu et l'obtenu et est donc la dernière couche de l'architecture. Parmi les différentes fonctions de perte existante, celle correspondant au cas d'application du projet est la perte par entropie croisée sigmoïde.

6.9 Exemple de modèle[6]

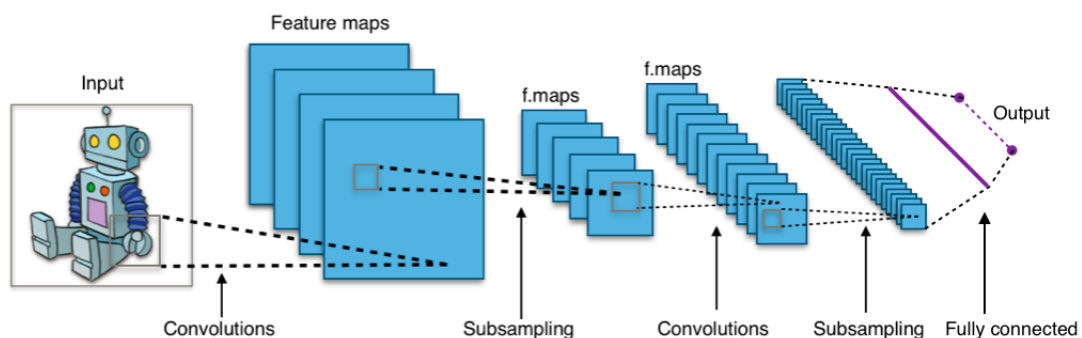


Figure 11 – Architecture standard d'un réseau à convolutions

Voici quelques architectures communes de réseau de neurones convolutifs qui suivent ce modèle :

- INPUT -> FC implémente un classifieur linéaire
- INPUT -> CONV -> RELU -> FC
- INPUT -> [CONV -> RELU -> POOL] * 2 -> FC -> RELU -> FC Ici, il y a une couche de CONV unique entre chaque couche POOL
- INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL] * 3 -> [FC -> RELU] * 2 -> FC Ici, il y a deux couches CONV empilées avant chaque couche POOL.

L'empilage des couches CONV avec de petits filtres de pooling (plutôt un grand filtre) permet un traitement plus puissant, avec moins de paramètres. Cependant, avec l'inconvénient de demander plus de puissance de calcul (pour contenir tous les résultats intermédiaires de la couche CONV).

4

Analyse et conception

1 Bibliothèques utilisées

Voici une liste des bibliothèques déjà utilisée par l'existant :

- OpenCV : pour le traitement des images
- Matplotlib : tracer toutes les figures.
- Scikit-learn : une bibliothèque gratuit d'apprentissage pour le langage de programmation Python
- Scikit-image : une bibliothèque de traitement d'image open source pour le langage de programmation Python
- NumPy : une extension du langage de programmation Python, ajoutant un support pour des tableaux et des matrices multidimensionnels, ainsi qu'une grande bibliothèque de fonctions mathématiques de haut niveau pour fonctionner sur ces tableaux.
- SciPy : une bibliothèque open source Python utilisée pour l'informatique scientifique et l'informatique technique.
- Sphinx : pour générer des documentations.
- Treelib : permet de construire des arbres, utilisée pour la gestion de l'arborescence de dossiers requise pour le fonctionnement du logiciel.

2 Diagramme de classes

Un diagramme de classe complet et maintenu à jour à été créé lors de la phase de recherche de ce projet. Trop volumineux pour être affiché ici, il est possible de le consulter à ce lien [lien](#) (des portions de ce diagramme seront utilisé pour illustrer certaines parties de ce rapport).

Deuxième partie

Développement

5

Découpage du projet en tâches

	Tâche	Durée	Début	Fin
1	Développement du projet	84 jours	10.01.18	04.04.18
2	Gestion de Workspaces	12 jours	10.01.18	21.01.2018
3	Conception	1 jour	10.01.2018	10.01.2018
4	Élaboration des tests	4 jours	11.01.2018	14.01.2018
5	Développement	5 jours	15.01.2018	19.01.2018
6	Documentation	2 jours	20.01.2018	21.01.2018
7				
8	Gestion des features	22 jours	22.01.18	12.02.2018
9	Analyse de l'existant	4 jours	22.01.2018	25.01.2018
10	Conception	7 jours	25.01.2018	31.01.2018
11	Élaboration des tests	3 jours	01.02.2018	03.02.2018
12	Développement	2 jours	04.02.2018	05.02.2018
13	Test de validation	3 jours	06.02.2018	09.02.2018
14	Rédaction documentation	3 jours	10.02.2018	12.02.2018
15	Gestion des résultats	13 jours	13.02.2018	25.02.2018
16	Analyse de l'existant	2 jours	13.02.2018	14.02.2018
17	Conception	3 jours	15.02.2018	17.02.2018
18	Développement	2 jours	18.02.2018	19.02.2018
19	Test de validation	3 jours	20.02.2018	22.02.2018
20	Rédaction documentation	3 jours	23.02.2018	25.02.2018
21				
22	Refonte IHM	13 jours	07.02.2018	09.03.2018
23	Élaboration mock up avec la MOA	1 jour	07.02.2018	07.02.2018
24	Validation par la MOA	0 jours	08.02.2018	25.02.2018
25	Conception	2 jours	26.02.2018	27.02.2018
26	Développement	3 jours	28.02.2018	02.03.2018
27	Test de validation	4 jours	03.03.2018	06.03.2018
28	Documentation	3 jours	07.03.2018	09.03.2018
29	Rédaction	24 jours	10.03.2018	04.04.2018
30	Rédaction du rapport	24 jours	10.03.2018	04.04.2018
31	Élaboration de la présentation	10 jours	26.03.2018	04.04.2018

Figure 1 – Diagramme de Gantt

Afin de mener à bien ce projet, les différentes fonctionnalités à implémenter ont été divisées en un certain nombre de tâches plus petites et permettre ainsi de mettre en exergue très facilement

les retards et avances éventuels. De plus, ce planning prend en compte qu'une première version doit être livrée à la MOA avant le 20 mars car ces derniers devront présenter cet outil lors d'une conférence.

1 Sprint 1 : Prise en main du projet

1.1 Description de la tâche

En l'absence de documentation existante, il est important dans un premier temps d'étudier le code afin de comprendre la conception de développeur précédent. Cette analyse est essentiel pour pouvoir retoucher à l'existant.

1.2 Livrable

Une documentation complète du code sera fournie à l'aide de la bibliothèque Sphinx. De plus, un rapport pour permettre l'installation du projet sera rédigé pour les développeurs qui reprendrai ce projet dans les prochaines années. Enfin, un diagramme de classe sera élaboré et validé par le tuteur académique.

1.3 Estimation de charge

Cette tâche est estimée à 10 jours/homme.

2 Sprint 2 : Workspace

2.1 Description de la tâche

Cette tâche permettra d'implémenter la notion de workspace définissant le répertoire dans lequel l'utilisateur travaillera. Cet élément correspondra au conteneur de notre modèle dans cette application. Cette classe sera évidemment retouché au cours de tout le développement de ce fait.

2.2 Livrable

Le projet sera livré avec la possibilité de créer un workspace en ligne de commande à partir d'un chemin d'accès vers un répertoire. L'arborescence requise par notre logiciel devra être créée si elle n'existe pas ; les noms des classes à reconnaître seront récupérés ; les noms des images d'entraînement seront également récupérés et un CSV contenant tous les pixels labellisés sera généré. Si une image binaire est manquante dans les sources fournies (voir page 6), alors le workspace devra créer une image blanche de même taille que l'image source.

2.3 Estimation de charge

Cette tâche est estimée à 10 jours/homme.

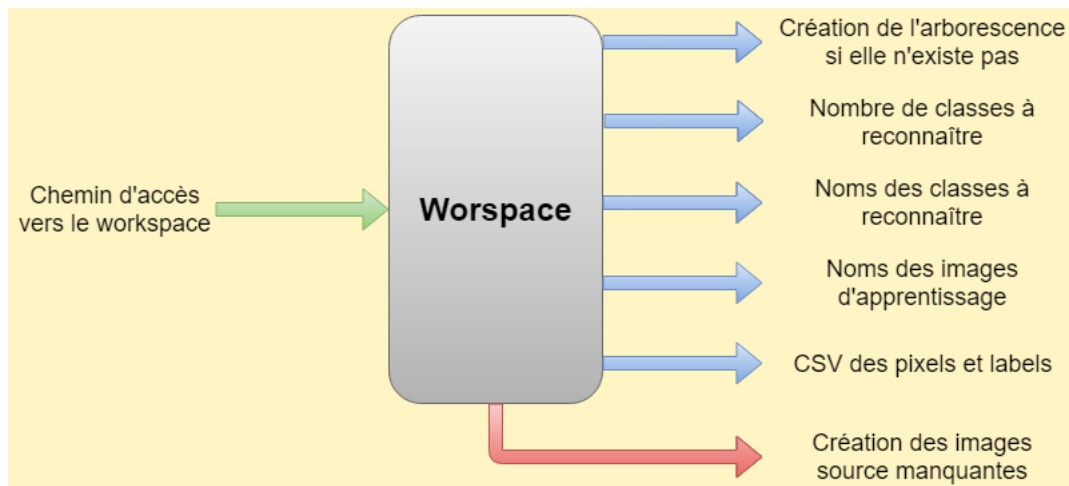


Figure 2 – Récapitulatif des fonctionnalités du workspace

3 Sprint 3 : Sélection des caractéristiques à exploiter

3.1 Description de la tâche

Actuellement, le modèle de reconnaissance utilise toutes les caractéristiques calculables pour s'entraîner. Or, il est fort probable que certaines d'entre elles sont inutiles, ou pire, induise le modèle en erreur. C'est pourquoi il est important que le logiciel permette à l'utilisateur de sélectionner les caractéristiques qu'il souhaite utiliser pour son modèle. La réduction du nombre de caractéristiques pourrait permettre à la fois d'augmenter la rapidité de l'exécution d'une prédiction d'images, puisqu'il y aurait moins de caractéristiques à calculer, mais surtout pourrait rendre le modèle plus efficace.

3.2 Livrable

Le projet sera livré avec la possibilité de prédire une image en passant en paramètre une liste des caractéristiques à utiliser.

3.3 Estimation de charge

Cette tâche est estimée à 15 jours/homme car elle nécessite de modifier de nombreuses classes.

4 Sprint 4 : Modification des images de résultats

4.1 Description de la tâche

Actuellement, le logiciel génère les résultats suivants lors d'une prédiction d'image :

- Une image binaire pour chaque classe à reconnaître : un pixel noir signifie que le modèle l'a considéré comme appartenant à la classe. Ces images correspondent aux images binaires d'entraînement.

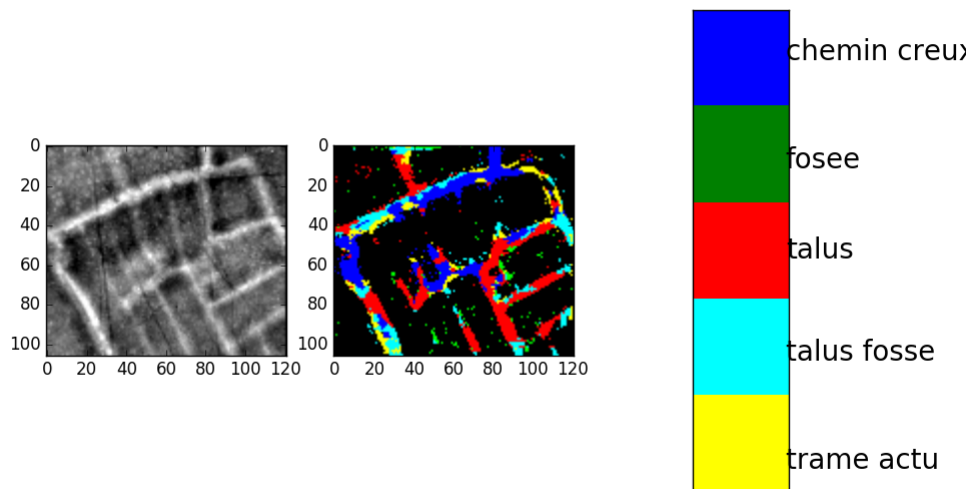


Figure 3 – Image comparative actuelle

- Une image comparant l'image source (en niveaux de gris) et une image où chaque couleur correspond à une classe.

La MOA souhaite pouvoir obtenir une image par classe en niveaux de gris correspondant à la **probabilité** d'appartenance à la classe. Plus le pixel sera foncé, plus il aura de chance qu'il appartienne à cette classe. De plus, l'image comparative doit être retravaillée.

4.2 Livrable

A partir de logiciel existant, la prédiction générera des images qui correspondront mieux aux attentes de la MOA.

4.3 Estimation de charge

Cette tâche est estimée à 10 jours/homme.

5 Sprint 5 : Mise en place de l'IHM

5.1 Description de la tâche

Refonte total de l'IHM nécessitant un travail en amont sur la conception existante. Tout d'abord, il faudra valider une MockUp avec la MOA avant de pouvoir l'implémenter. Cette tâche doit être faite en dernier car elle dépendra grandement de la conception globale du système.

Parmis les éléments souhaités, on peut noter l'utilisation de plusieurs vues afin de mieux guider l'utilisateur ; L'utilisation de threads afin que l'application continue de répondre durant de long calculs ; la présence de barres de progressions afin d'informer l'utilisateur de l'avancée de son processus.

5.2 Livrable

Le produit dans sa globalité avant la date butoir du 20 mars.

5.3 Estimation de charge

Cette tâche est estimée à 30 jours/homme.

6 Sprint 6 : Rédaction du rapport

6.1 Description de la tâche

Cette tâche consiste à la rédaction du rapport final . Bien entendu, il sera tout de même travaillé tout au long du projet.

De plus, cette période sera utilisée pour corriger les problèmes éventuels relevés par la MOA et l'élaboration de tests unitaires et de validation.

6.2 Livrable

Le rapport final.

6.3 Estimation de charge

Cette tâche est estimée à 15 jours/homme.

6

Implémentation

Dans ce chapitre sera décrit comment les différentes tâches listées ci-dessus ont été implémentées. Il permettra donc de témoigner de la qualité du produit rendu et pourra aiguiller les éventuels étudiants reprenant ce projet dans les années suivantes.

1 Analyse de l'existant

Cette phase de développement fut commencé par une analyse approfondie de l'existant. En l'absence de documentation, il a fallut comprendre la conception pour mettre en exergue ces qualités ainsi que ces faiblesses.

1.1 Variables statiques

Voici ce que nous pouvions lire dans un fichier source nommé config.py :

```
1 CLASS_NAMES = ['C1', 'C2', 'C3', 'C4', 'C5']
2
3 # class names of the model
4 REAL_CLASS_NAMES = ['chemin_creux', 'fosee', 'talus', 'talus_fosse', 'trame_actu']
5 # colors : http://www.rapidtables.com/web/color/RGB\_Color.htm
6 # ['blue', 'green', 'red', 'cyan', 'yellow'])
7 CLASS_COLORS = [[0,0,255], [0,255,0], [255,0,0], [0,255,255], [255,255,0]]
8
9 # total : 17 features
10 # FEATURE_NAMES = ['Gray',
11 #                   'Avg3x3', 'Avg5x5', 'Avg9x9', 'Avg15x15', 'Avg19x19',
12 #
13 #                   'MinD3x3', 'MaxD3x3', 'AvgD3x3',
14 #                   'MinD5x5', 'MaxD5x5', 'AvgD5x5',
15 #                   'MinD9x9', 'MaxD9x9', 'AvgD9x9',
16 #                   'MinD15x15', 'MaxD15x15', 'AvgD15x15',
17 #                   'MinD19x19', 'MaxD19x19', 'AvgD19x19',
18 #
19 #                   'NumB3x3', 'NumD3x3', 'NumF3x3', 'NumW3x3',
20 #                   'NumB5x5', 'NumD5x5', 'NumF5x5', 'NumW5x5',
```

```

21 #             'NumB9x9', 'NumD9x9', 'NumF9x9', 'NumW9x9',
22 #             'NumB15x15', 'NumD15x15', 'NumF15x15', 'NumW15x15',
23 #             'NumB19x19', 'NumD19x19', 'NumF19x19', 'NumW19x19'
24 #         ]
25
26 FEATURE_NAMES = [ 'Gray',
27                   'Avg3x3', 'Avg5x5', 'Avg9x9', 'Avg15x15', 'Avg19x19',
28
29                   'MinD3x3', 'MaxD3x3', 'AvgD3x3',
30                   'MinD5x5', 'MaxD5x5', 'AvgD5x5',
31                   'MinD9x9', 'MaxD9x9', 'AvgD9x9',
32                   'MinD15x15', 'MaxD15x15', 'AvgD15x15',
33                   'MinD19x19', 'MaxD19x19', 'AvgD19x19',
34
35                   'NumB3x3', 'NumD3x3', 'NumF3x3', 'NumW3x3',
36                   'NumB5x5', 'NumD5x5', 'NumF5x5', 'NumW5x5',
37                   'NumB9x9', 'NumD9x9', 'NumF9x9', 'NumW9x9',
38                   'NumB15x15', 'NumD15x15', 'NumF15x15', 'NumW15x15',
39                   'NumB19x19', 'NumD19x19', 'NumF19x19', 'NumW19x19'
40 ]

```

L'utilisation de variables statiques est très souvent mauvais signe. La variable `CLASS_NAMES` signifie que les images sources doivent être stockées dans des répertoires nommés C1, C2, C3, C4 et C5. La variable `REAL_CLASS_NAMES` signifie, quant à elle, que le logiciel peut reconnaître cinq, et uniquement cinq classes différentes nommées `chemin_creux`, `fosee`, `talus`, `talus_fosse` et `trame_actu`. Cela témoigne de l'absence d'évolutivité dans le système.

De plus, la variable `FEATURE_NAMES` signifie que les features à extraire sont constantes et ne peut donc être modifiées.

Pour résumé, l'existant manque cruellement d'évolutivité, ce qu'il faudra résoudre, et ce genre de variable statique est à banir.

1.2 Convention de nommage

Le langage Python a la particularité d'utiliser la même convention de nommage snake case, contrairement à la plupart des autres langages informatiques. Voici plusieurs exemple de comment écrire différents nom de variable :

- "nom de variable" devient `nom_de_variable`
- "NomDeVariableCamelCase" devient `nom_de_variable_camel_case`
- "Variable" devient `variable`
- "variable" devient `variable`

Cette convention n'est pas respecté dans ce projet, parfois nous retrouvons du snake case, mais la plupart du temps du camel case. Dasn un but de lisibilité du code, les noms de variables devront être modifiés.

Avec l'IDE PyCharm que nous utilisons dans ce projet, il sera très d'effectuer cette opération.

1.3 Propriétés en Python

Le Python est un langage pour sa lisibilité, il est donc conçu pour que l'on est le moins de choses à écrire. C'est pourquoi, en Python, il n'existe pas de propriété privée et donc, il n'y aura jamais

besoin d'utiliser de getter ou de setter sauf dans le cas où vous souhaitez effectuer une opération spécial lors d'un get ou d'un set.

Le code existant présentant certains getter et setter, il devra être épuré afin de toujours garder une lisibilité optimale.

2 Workspace

Après une analyse approfondie de l'existant et une phase de conception, voici la classe Workspace qui fut implémentée :

Workspace
<pre> - required_tree: Tree - class_names: str[] - directory_path: str - training_image_names: str[] - faithful: boolean - recognition_model: RecognitionModel - learning_data_set: FeatureTable - evaluate_data_set: FeatureTable - recognition_model: RecognitionModel - extracted_features: Feature[] - selected_features: Feature[] - result_builders: ResultBuilder[] </pre>
<pre> + find_class_names(): str[] + find_training_images_names(): str[] + init(filename: str): void + is_faithful(dirpath: str, filename: str): boolean + create_required_structure(): void + update(): void + generate_csv_dictionary(): void + extract_validation_data_set_from_learning_data_set(nb_samples_per_classes: int): void + extract_training_images_features(): void + train(nb_samples_per_classes: int = 100, nb_samples_for_evaluation: int = 10): void + evaluate(): Evaluation + predict(path_image: str, image_names: [str]): void + generate_results(image_name: str): void + set_features_list(feature_names: str[]): void + find_features_already_extracted(nb_samples_per_classes: int): [DefinitionFeature] </pre>

Figure 1 – Classe Workspace

2.1 Evolutivité de l'arborescence requise

Tout d'abord, la constante *required_tree* permet de définir, sous forme d'un arbre, l'arborescence nécessaire au bon fonctionnement de l'application. L'utilisation de cette constante permet une grande évolutivité, si jamais nous souhaitons modifier un jour cette arborescence. Pour créer cette constante, nous utilisons la bibliothèque TreeLib. Le seul inconvénient de cette solution est unicité des noms de dossiers.

Cette constante sera ensuite utilisée dans les méthodes `is_faithful` (permettant de vérifier si l'arborescence est conforme) et `create_required_structure` (permettant de créer les dossiers manquants dans le cas où l'arborescence n'est pas conforme).

```

1  # Definition of our required repository structure, this object will be used in the
    following method to verify
2  # if the workspace is faithful or not.
3  #
4  # Root
5  # |-- features
6  # |   |-- predict
7  # |   |-- train_features
8  # |-- machinelearning
9  # |   |-- evaluate
10 # |   |-- results
11 # |   |-- svm
12 # |   |-- train_machinelearning
13 # |-- predict-images
14 # |-- train-images
15 #     |-- img
16 required_tree = Tree()
17 required_tree.create_node("Root", "Root")
18 required_tree.create_node("features", "features", parent="Root")
19 required_tree.create_node("predict", "predict", parent="features")
20 required_tree.create_node("train_features", "train_features", parent="features")
21 required_tree.create_node("machinelearning", "machinelearning", parent="Root")
22 required_tree.create_node("evaluate", "evaluate", parent="machinelearning")
23 required_tree.create_node("results", "results", parent="machinelearning")
24 required_tree.create_node("svm", "svm", parent="machinelearning")
25 required_tree.create_node("train_machinelearning", "train_machinelearning", parent="
    machinelearning")
26 required_tree.create_node("predict-images", "predict-images", parent="Root")
27 required_tree.create_node("train-images", "train-images", parent="Root")
28 required_tree.create_node("img", "img", parent="train-images")

```

2.2 Parcours de l'arborescence

Une fois l'arborescence créée ou validée, le workspace utilise le module `os` de Python pour récupérer les noms des dossiers afin d'enregistrer les informations suivantes :

- Les noms des classes à reconnaître (`find_class_names()`) : Les noms des dossiers présents dans le répertoire `train-images` (`img` exclu). Ce sont les dossiers dans lesquels seront stockées les images binaires d'entraînements.
- Les noms des images d'entraînements (`find_training_images_names()`) : Les noms des images sources qui ont été stockées dans le répertoire `train-images/img`. Ce sont les images d'entraînement.

Ces deux fonctions sont très importantes car elles permettent au logiciel d'avoir la souplesse manquante à cause de l'utilisation des constantes comme expliqué page 24.

3 Sélection des caractéristiques à exploiter

Afin de permettre au système de sélectionner quelles caractéristiques il faut exploiter, nous sommes obligés de modifier la conception de la classe `Feature` et `ImageFeaturesExtractor`.

ImageFeaturesExtractor
+ all_definition_features: DefinitionFeature[]
+ extract_all_features(image_path: str, labeled_pixels: LabeledPixel[], nb_random_pixel_to_add: int): FeatureTable
+ extract_features(image_path: str, features: [DefinitionFeature], labeled_pixels: [LabeledPixel] = None, nb_random_pixel_to_add: int = 0): FeatureTable
+ generate_training_base(individuals_csv_files_path: str, file_names: [str], nb_samples_per_classes: int, nb_classes: int, list_features: [DefinitionFeature]): FeatureTable

Figure 2 – Classe `ImageFeaturesExtractor`

3.1 Constante `all_definition_features`

Afin de récupérer toutes les features existantes, il est obligatoire de faire appel à une nouvelle constante, mais elle diffère avec celle déjà existante de part sa création qui se base sur une énumération et permet donc d'être automatiquement mise à jour lorsque de nouvelles features sont implémentées.

```

1  # The different sizes we will use
2  all_mask_size = [3, 5, 9, 15, 19, 25, 35]
3  all_definition_features = set()
4
5  # For all the features available
6  for definition_feature in DefinitionFeatureEnum:
7      if definition_feature.need_mask_size:
8          for mask_size in all_mask_size:
9              all_definition_features.add(DefinitionFeature(definition_feature, mask_size))
10     else:
11         all_definition_features.add(DefinitionFeature(definition_feature))

```

3.2 Conception de la classe `Feature`

L'existant contenait une classe `Feature`, abstraite, contenant une méthode abstraite `compute`, implémenté par chacune des sous-classes de `Feature`.

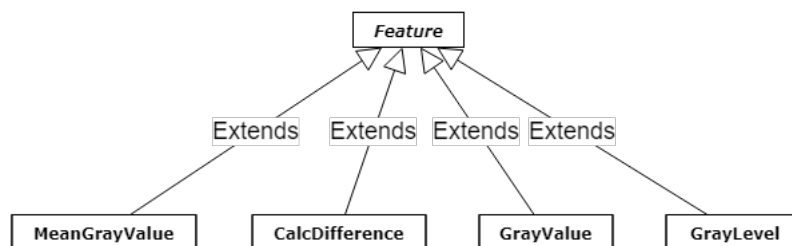


Figure 3 – Conception des classes de features de l'existant

Cette conception est une très bonne idée et est à retenir. Cependant, les différentes sous-classes ne fonctionnent pas de la manière. En effet, il est parfois possible de calculer plusieurs features

en même temps est donc le retour de la fonction n'était pas toujours du même type, ce qui est un défaut de conception possible uniquement parce que le langage Python doté d'un typage dynamique fort.

```

1 class GrayValue(Feature):
2
3     def compute(self, pixel, img=None):
4         if img is None:
5             image = self.getImg()
6         else:
7             image = img
8
9         return image.getPixelGrayValue(pixel)

```

```

1 class CalcDifference(Feature):
2     def compute(self, pixel, img=None):
3
4         # [...]
5
6         if not roi.size == n * n:
7             # maybe this is not the best solution to process corner pixel
8             return 0, 0, 0
9         else:
10            central_value = roi[(n-1)//2][(n-1)//2]
11            diff = list()
12            for v in np.nditer(roi):
13                diff.append(abs(int(central_value) - int(v)))
14            # remove the central_value - central_value
15            del diff[(n*n - 1)//2]
16            # return the min/max/avg of the differences.
17            return min(diff), max(diff), sum(diff)/len(diff)

```

Dans l'exemple ci-dessus, nous pouvons voir que la classe `GrayValue` nous renverra un entier tandis que la classe `CalcDifference` renverra un tuple de taille 4. Étant donné que le comportement n'est pas le même suivant la classe instanciée, nous perdons ici notre abstraction.

Afin de remédier à ce problème, nous avons dû faire en sorte que chaque sous-classe renvoie un dictionnaire, c'est-à-dire un couple clé, valeur avec comme valeur la mesure et calculée et comme clé, le nom de la caractéristique.

De plus, nous devons faire une différence entre une `Feature` et une `DefinitionFeature`. Si la `Feature` calcul une ou des caractéristiques, la `DefinitionFeature` correspond à une seule caractéristique est définie quelle est la `Feature` à utiliser pour la calculer.

Voici la nouvelle version de nos sous-classes `Feature` :

```

1 class GrayValue(Feature):
2     """ This class represents a feature: the gray scale of the pixel.
3     """
4
5     def compute(self, pixel: (int, int), img, mask_size: int = None):
6         """ The core function to compute the the feature value of a given pixel.
7
8         :param pixel: the pixel, obligatory
9         :type pixel: (int, int)

```

```

10 :param img: the image, obligatory
11 :type img: OpenCV Image
12 :param mask_size: the size of neighborhood of the pixel to compute feature
    value, optional
13 :type mask_size: int
14 :return: A dictionary containing the features computed.
15 :rtype: {str, float}
16 """
17 return dict([(self.name, img.get_pixel_gray_value(pixel))])

```

```

1 class CalcDifference(Feature):
2     """ This feature returns the minimum of the differences between central value
3         and neighborhood values in n * n mask.
4     """
5
6     def compute(self, pixel: (int, int), img, mask_size: int = None):
7         """ The core function to compute the the feature value of a given pixel.
8
9             :param pixel: the pixel, obligatory
10            :type pixel: (int, int)
11            :param img: the image, obligatory
12            :type img: OpenCV Image
13            :param mask_size: the size of neighborhood of the pixel to compute feature
                value, optional
14            :type mask_size: int
15            :return: A dictionary containing the features computed.
16            :rtype: {str, float}
17
18            :raise AttributeError: If the msk_size isn't an odd number
19            """
20            if mask_size is None or mask_size % 2 == 0:
21                raise AttributeError('The mask_size must be an odd number')
22
23            roi = img.get_region_of_image(pixel, mask_size)
24
25            result = dict()
26            if not roi.size == mask_size * mask_size:
27                # maybe this is not the best solution to process corner pixel
28                result[self.construct_full_name(MIN_CALC_DIFF_NAME, mask_size)] = 0
29                result[self.construct_full_name(MAX_CALC_DIFF_NAME, mask_size)] = 0
30                result[self.construct_full_name(AVG_CALC_DIFF_NAME, mask_size)] = 0
31            else:
32                central_value = roi[(mask_size - 1) // 2][(mask_size - 1) // 2]
33                diff = list()
34                for v in np.nditer(roi):
35                    diff.append(abs(int(central_value) - int(v)))
36                # remove the central_value - central_value
37                del diff[(mask_size * mask_size - 1) // 2]
38                # return the min/max/avg of the differences.
39                result[self.construct_full_name(MIN_CALC_DIFF_NAME, mask_size)] = min(diff)
40                result[self.construct_full_name(MAX_CALC_DIFF_NAME, mask_size)] = max(diff)
41                result[self.construct_full_name(AVG_CALC_DIFF_NAME, mask_size)] = sum(diff)
42                    / len(diff)
43            return result

```

3.3 Evolutivité des features

Afin de démontrer en quoi cette nouvelle conception est évolutive, voici les étapes à suivre pour ajouter une nouvelle feature avec un exemple concret, le local binary patterns, ajouté dans le système suite à une demande de la MOA.

Brièvement, cette caractéristique correspond à de la reconnaissance de motif, c'est-à-dire que les pixel ayant la même valeur sont inscrits dans le même motif.



Figure 4 – Exemple de local binary patterns

Tout d'abord, il faut créer une nouvelle classe héritant de la classe abstraite Feature et donc définir sa fonction compute.

Ensuite, il faut ajouter une ligne à l'énumération DefinitionFeatureEnum :

```

1 LOCAL_BINARY_PATTERN_NAME = 'LBP'
2
3 class DefinitionFeatureEnum(Enum):
4     """This enumeration is used to retrieve all the possible features. A definition
5         feature enumeration contains :
6
7         * A name which must be unique
8         * The feature to compute to retrieve his value
9         * A boolean to say if you need a size or not
10
11     """
12     # [...]
13
14     LOCAL_BINARY_PATTERN_NAME_ENUM = (LOCAL_BINARY_PATTERN_NAME, LocalBinaryPatterns(
15         LOCAL_BINARY_PATTERN_NAME), True)

```

4 Modification des images de résultats

4.1 Nouvelle conception

Afin d'assurer l'évolutivité à nouveau, une classe abstraite ResultBuilder fut créé. Jusqu'à présent, les résultats étaient créés dans une seule fonction construct_results.

Grâce à cette nouvelle conception, l'ajout d'un nouveau moyen de stocker les résultats se fera de la manière la plus simple possible : il suffira de créer une classe héritant de la classe abstraite ResultBuilder et de lui définir sa méthode generate_results. Tout le reste se fera automatiquement.

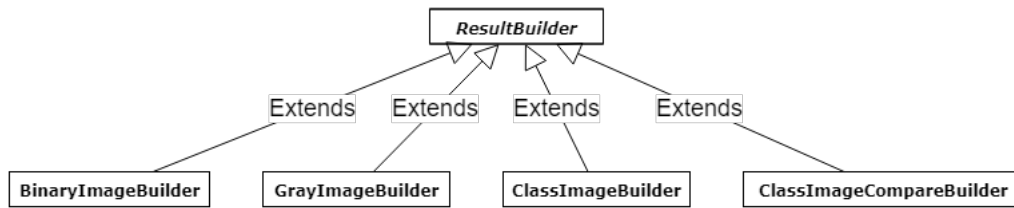


Figure 5 – Nouvelle conception des résultats

4.2 Modification de l'image comparative

Maintenant que nous avons permis au système d'avoir un nombre de classes à reconnaître variable ainsi que leur noms, il faut remettre à jour la génération de l'image comparative.

Pour générer des couleurs les plus différentes possibles, nous utilisons le système de gestion des couleurs HSV (Hue Saturation Value) car parmi ses trois composantes, celle qui correspond à la teinte se trouve correspondre à un angle. Il suffit donc de diviser 360 par le nombre de couleurs à générer pour obtenir les couleurs les plus éloignées possibles.

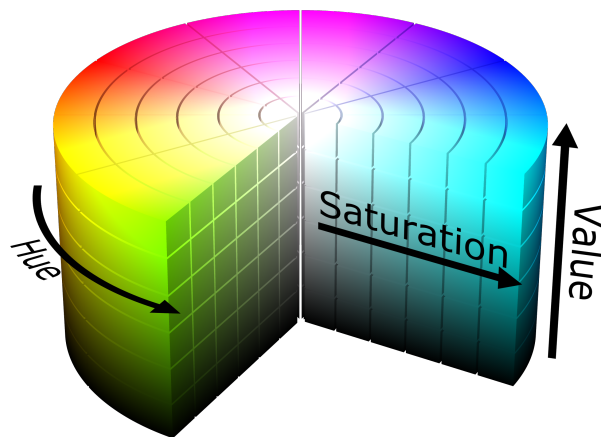


Figure 6 – Système de gestion des couleurs HSV

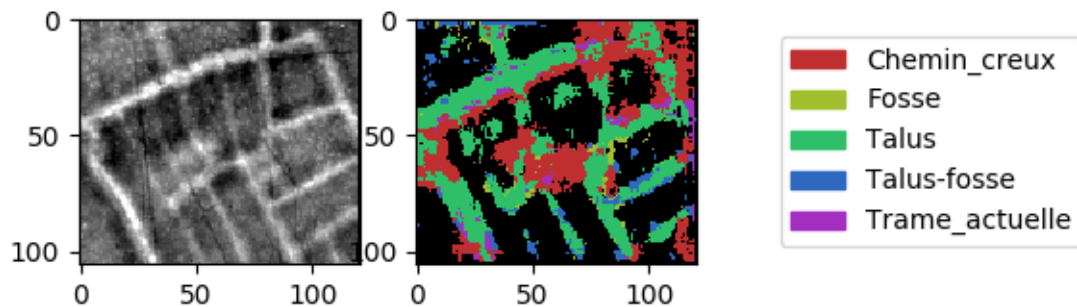


Figure 7 – Nouvelle version des résultats

5 Mise en place de l'IHM

5.1 Élaboration de la MockUp

A l'aide de l'outil en ligne draw.io, nous avons pu dessiner avec la MOA une MockUp qui pouvait, tout comme un Google doc, être modifié par plusieurs acteurs simultanément.

Une fois la MockUp validé par le client, nous n'avons plus touché à l'interface et avons pu rentrer dans notre phase d'implémentation de l'IHM.

5.2 Implémentation des vues

Afin de créer les vues le plus simplement possible, nous avons utilisé QtDeveloper, un logiciel permettant de créer des vues uniquement en faisant du drag-and-drop d'élément. Cet outil très visuel stocke ces production au format .ui, il faut ensuite utiliser un script Pyqt afin de traduire ces vues en Python.

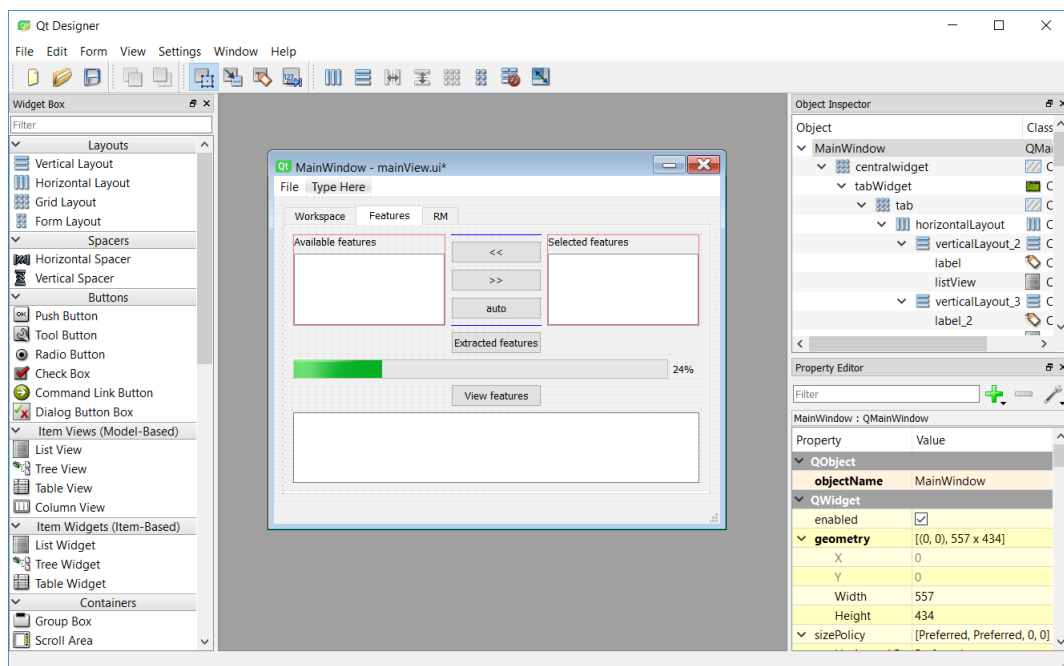


Figure 8 – QtDesigner, outil de création d'interfaces graphiques

5.3 Vues créées

A chaque fois qu'un processus potentiellement long est demandé par l'utilisateur, un thread est lancé pour effectuer ces calculs. Cela nous permet d'éviter le fameux "Ne réponds pas" que nous avons déjà tous constaté lorsqu'un logiciel ne fonctionne plus.

La première vue est celle qui apparaît au lancement de l'application. Elle permet de choisir le dossier qui correspondra au workspace dans lequel nous travaillerons.

Ensuite, la vue principale, composée de trois onglets différents, permet de consulter les noms des classes à reconnaître et les noms des images d'entraînement, de sélectionner et d'extraire les

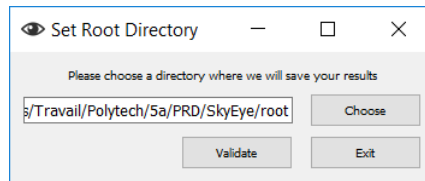


Figure 9 – Vue de démarrage

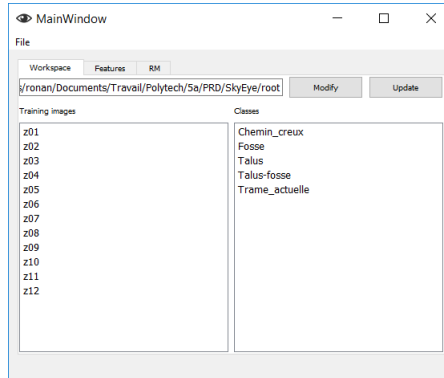


Figure 10 – Vue principale reprenant le nom des classes et d'images d'entraînement

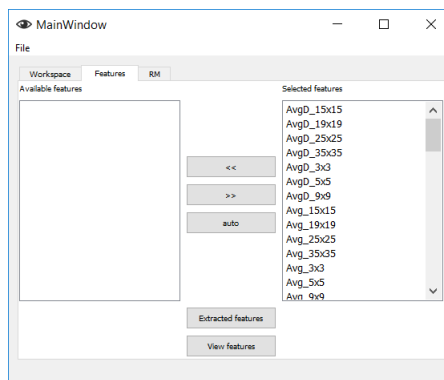


Figure 11 – Tab permettant d'extraire les features

caractéristiques de notre choix, d'entraîner et d'entraîner notre modèle et enfin de prédire une image.

Dans l'onglet concernant le modèle, un cadre est réservé aux paramètres du modèle. Celui-ci a été programmé de manière à ce qu'il s'adapte automatiquement avec de nouveaux paramètres le jour où d'autres types de modèle de reconnaissance seront implémentés.

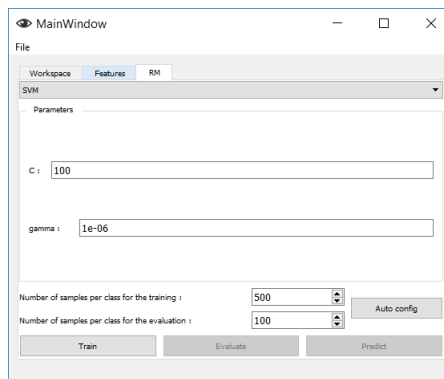


Figure 12 – Tab permettant d'interagir avec le modèle de reconnaissance

Lors d'une évaluation, une nouvelle vue est affichée afin d'obtenir le taux de reconnaissance du modèle ainsi que sa matrice de confusion. En ligne, nous avons les classes que nous avons données au modèle, et en colonne, sa réponse.

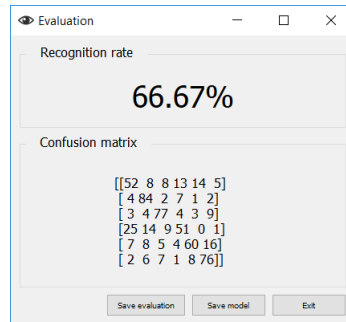


Figure 13 – Vue montrant les résultats d'une évaluation

7

Qualité mise en œuvre

1 Documentation

Afin de permettre d'éventuels étudiants de reprendre ce projet, de nombreux documents ont été rédigés :

- La documentation du code à l'aide de la librairie Sphinx
- Un guide utilisateur permettant d'installer l'environnement et de comprendre comment utiliser le logiciel
- Un guide développeur contenant des conseils pour les développeurs

De plus, les différents diagrammes de classes et de cas d'utilisation sont versionnés sur le git au format .xml et peuvent donc être exploités à nouveau.

2 Tests unitaires

En plus des différents documents rédigés, il est primordial de mettre en place de nombreux tests afin d'assurer la qualité du code produit. C'est pourquoi ce projet comporte des tests unitaires.

De plus, des tests de validation ont été effectués à la fin de la phase de développement pour vérifier qu'une régression n'est été faite.

Malheureusement, le manque de temps nous a trop souvent obligé à privilégier la production de nouvelles fonctionnalités plutôt que la mise en place de batteries de test.

Les tests unitaires ont été à l'aide de la librairie Python unittest. En voici un exemple :

```
1 class TestWorkspace(TestCase):
2     def __init__(self, methodName='runTest'):
3         super().__init__(methodName)
4         self.fairful_path = join(os.getcwd(), "FairfulWorkspace")
5         self.not_fairful_path = join(os.getcwd(), "NotFairfulWorkspace")
6         self.not_fairful_empty_path = join(os.getcwd(), "NotFairfulEmptyWorkspace")
7         self.fairful_workspace = Workspace(self.fairful_path)
8         self.not_fairful_workspace = Workspace(self.not_fairful_path)
9         self.not_fairful_empty_workspace = Workspace(self.not_fairful_empty_path)
10
11     def setUp(self):
```

```

12     super().setUpClass()
13     shutil.rmtree(self.not_faithful_empty_path)
14     os.makedirs(self.not_faithful_empty_path)
15     if os.path.exists(join(join(self.not_faithful_path, "machinelearning"), "svm"))
16         :
17         shutil.rmtree(join(join(self.not_faithful_path, "machinelearning"), "svm"))
18
19 def test_is_faithful(self) -> None:
20     self.assertTrue(self.faithful_workspace.is_faithful())
21     self.assertFalse(self.not_faithful_workspace.is_faithful())
22     self.assertFalse(self.not_faithful_empty_workspace.is_faithful())
23     return
24
25 def test_create_required_structure(self) -> None:
26     self.assertFalse(self.not_faithful_workspace.is_faithful())
27     self.assertFalse(self.not_faithful_empty_workspace.is_faithful())
28     self.not_faithful_workspace.create_required_structure()
29     self.not_faithful_empty_workspace.create_required_structure()
30     self.assertTrue(self.not_faithful_workspace.is_faithful())
31     self.assertTrue(self.not_faithful_empty_workspace.is_faithful())
32     return
33
34 def test_find_class_names(self) -> None:
35     self.assertEqual(self.faithful_workspace.find_class_names(), ['C1', 'C2', 'C3',
36         'C4', 'C5'])
37
38     with self.assertRaises(AttributeError):
39         self.not_faithful_workspace.find_class_names()
40     return
41
42 def test_find_training_image_names(self) -> None:
43     self.assertEqual(self.faithful_workspace.find_training_image_names(),
44         ['z01', 'z02', 'z03', 'z04', 'z05', 'z06', 'z07', 'z08'])
45
46     with self.assertRaises(AttributeError):
47         self.not_faithful_workspace.find_training_image_names()
48     return

```

3 Tests de validation

Pour mettre en place des tests de validation, une copie du projet initial fut fait en début de projet. De ce fait, il est possible de lancer le même processus avec l'ancienne est la nouvelle version. Il suffit ensuite de comparer les résultats.

Un résultat fut donc généré avec l'ancienne version et un workspace équivalent fut créé pour la version actuelle. Par contre, la comparaison ce fait pour l'instant à la main par manque de temps d'élaborer des tests de validation automatiques.

8

Récapitulatif

Voici un petit récapitulatif de tous ce qui a été apporté au logiciel durant ce projet.

	Modifications apportées
1	Evolutivité de l'arborescence requise
2	Evolutivité du nombre de classes à reconnaître
3	Evolutivité des classes à reconnaître
4	Possibilité de choisir les features à exploiter
5	Algorithme de sélection des meilleures features à utiliser
6	Nouvelle IHM utilisant des threads
7	Ajout d'un nouveau type de résultats
8	Renommage des variables suivant la convention de Python
9	Mise en place d'un pattern MVC
10	Élaboration de tests unitaires
11	Élaboration de tests de validation
12	Ajout d'une nouvelle feature : Local Binary Pattern
13	Rédaction d'un guide d'utilisation
14	Rédaction d'un guide du développeur
15	Rédaction d'une documentation du code
16	
17	
18	

Figure 1 – Récapitulatif du projet

9

Conclusion

Avec un total de 328 heures d'autonomies réservées dans notre emploi du temps pour ce projet depuis le mois de septembre, nous pouvons constater une nette progression sur ce projet.

En plus des améliorations citées plus haut, ce projet fut l'une des expériences les plus enrichissantes que j'ai pu vivre.

A cause de nombreuses circonstances, j'ai commencé mon PR&D avec un mois de retard et ce projet fut le premier que je fis avec un réel client, ce qui m'a appris à gérer les différents entités lié à un projet. En effet, le client souhaite essentiellement voir de nouvelles fonctionnalités tandis que le tuteur académique souhaite obtenir plus de documentation et d'éléments permettant d'assurer la qualité des travaux (conception, test, gestion de projet). De plus, les heures de projets ne suffisant pas, il est primordial de savoir gérer son temps afin de ne pas se laisser déborder ainsi que de s'organiser au mieux.

Ensuite, j'ai pu constater l'importance de la documentation du code lors de mon analyse de l'existant. En l'absence de cette dernière, j'ai perdu énormément de temps à comprendre ce qui été déjà implémenté. C'est pourquoi la documentation de mon code fut quelque chose que j'ai voulu mettre absolument en avant, ajouter des commentaires, générer une documentation et rédiger un guide d'installation.

Enfin, je peux terminer en disant que malgré l'utilisation d'un langage que je ne connaissais pas avant le début du projet, j'ai réussi à me former et à apporter une réelle plus value au logiciel. Je ne pensais pas être capable de faire tout ce que j'ai implémenté et en suis donc très fier, ce qui est pour moi le plus important dans un projet et dans une équipe de développement en général.

De nombreuses fois lors de notre formation d'ingénieur, nos enseignants nous expliquaient que Polytech Tours avait pour but de nous apprendre à apprendre car c'est là la plus grande qualité d'un ingénieur. Je suis donc heureux de constater que je suis désormais capable d'apprendre par moi même une technologie qui m'était totalement inconnue. Ce projet est donc une expérience parfaite pour conclure ces cinq années de formation à Polytech Tours, permettant à chacun d'entre nous de nous rendre compte que nous sommes effectivement prêt à démarrer notre vie professionnelle.

Annexes

A

Diagramme de cas d'utilisation

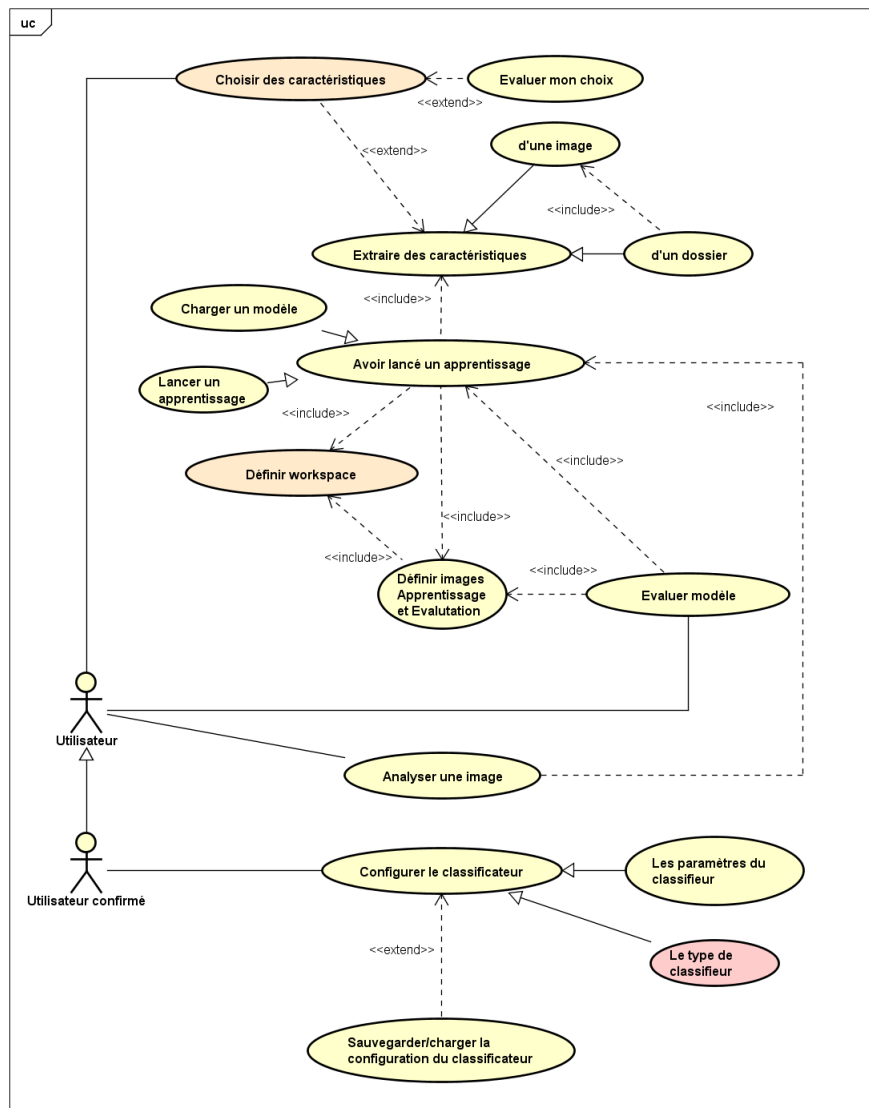


Figure 1 – Diagramme de cas d'utilisation

B

Cahier des spécifications

ÉCOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Spécialité Informatique

64 av. Jean Portalis

37200 TOURS, FRANCE

Tél +33 (0)2 47 36 14 31

www.polytech.univ-tours.fr

CAHIER DE SPECIFICATION

Projet : CDS40

Plateforme d'extraction et de caractérisation automatique d'éléments d'intérêt dans les images archéologiques par apprentissage interactif

Emetteur :

Ronan GUILLAUME

MOA :

Jean-Yves RAMEL
Clément LAPLAIGE
Xavier RODIER

Date d'émission :

26/10/2017

Validation

Nom	Date	Valide (O/N)	Commentaires

Historique des modifications

Version	Date	Description de la modification
0	26/10/2017	Création du document + Ebauche du plan
0.1	09/11/2017	Rédaction de l'introduction + Description générale
0.2	21/11/2017	Spécifications Fonctionnelles
1.0	29/11/2017	Fin première version
2.0	09/12/2017	Fin deuxième version

TABLE DES MATIERES

Cahier de spécifications	3
1. Introduction	3
1.1. Les acteurs du projet.....	3
1.2. Contexte de la réalisation	3
2. Description générale	5
2.1. Environnement du projet.....	5
2.2. Caractéristiques des utilisateurs	5
2.3. Fonctionnalités du système	6
2.4. Structure générale du système	6
2.5. Description des Interfaces homme/machine	9
3. Spécifications fonctionnelles.....	9
3.1. Gestion du workspace.....	9
3.2. Gestion de l'extraction des caractéristiques des images d'apprentissage.....	11
3.3. Gestion de l'extraction de caractéristiques	12
3.4. Gestion des classifieurs	13
3.5. Exploitation des résultats.....	14
4. Spécifications non fonctionnelles	15
4.1. Contraintes de développement et conception	15
4.2. Performances	15
5. Autres tâches importantes.....	15
5.1. Génération de documentations	15
5.2. Modification de l'IHM	15
6. Conclusion.....	15
Bibliographie	16

1. Introduction

Dans le cadre de la formation en ingénieur informatique à l'Ecole Polytechnique de Tours, un Projet de Recherche & Développement (PR&D) est réalisé en 5ème année afin de constituer une réelle expérience en terme de conduite de projet. Cette année, le PR&D se déroule du 11 septembre 2017 au 5 avril 2018 à raison de 2 jours par semaine à temps plein. Il donne lieu à la rédaction d'un rapport avec le cahier de spécification et de deux présentations du travail effectué lors de deux soutenances.

Le PR&D est orienté vers une conception logiciel de reconnaissance d'images où la MOA est représentée par Clément LAPLAIGE et Xavier RODIER, chercheurs en archéologie du laboratoire LAT - CITERES et Mr Jean-Yves RAMEL, enseignant chercheur à l'Ecole Polytechnique de Tours. Le projet s'inscrit dans un cadre théorique qui fait l'objet d'une étude algorithmique en amont, pour aboutir à l'amélioration d'un logiciel.

Ce document est donc le cahier de spécification système du projet « Plateforme d'extraction et de caractérisation automatique d'éléments d'intérêt dans les images archéologiques par apprentissage interactif ». Il définit les besoins, l'environnement du projet et les objectifs à réaliser. Ce rapport présente aussi les différentes tâches à effectuer et le planning prévisionnel.

1.1. Les acteurs du projet

– la maîtrise d'œuvre (MOE) : Ronan GUILLAUME, étudiant en 5ème année de l'Ecole d'Ingénieur Polytech Tours au sein du département Informatique, dans le cadre du PR&D ainsi que son encadrant Jean-Yves RAMEL de l'équipe de recherche RFAI - EA 6300 du Laboratoire Informatique de Tours (<http://www.rfai.li.univ-tours.fr>).



Figure 1 - Logo du groupe RFAI du Laboratoire Informatique de Tours

Les recherches menées au sein de cette équipe portent sur l'étude et la résolution des problèmes de reconnaissance d'images, que ce soit dans le cadre de problématiques "industrielles" ou de problèmes théoriques.

– la maîtrise d'ouvrage (MOA) : Le projet SOLiDAR du laboratoire Archéologique et Territoire CITERES de Tours, représenté par Clément LAPLAIGE et Xavier RODIER. Créé en 2004, ce laboratoire basé sur Tours a pour objectif d'analyser les dynamiques spatiales et territoriales des sociétés. Dans ce contexte, leur champ de recherche se compose de quatre secteurs : la recherche urbaine, la recherche environnementale, les travaux sur le territoire et ceux sur les effets des recompositions sociales contemporaines. (<http://citeres.univ-tours.fr>)



Figure 2 - Logo du projet SOLiDAR et du laboratoire CITERES de Tours

Ce document a été rédigé par Ronan GUILLAUME et sera validé par différents acteurs du projet.

1.2. Contexte de la réalisation

1.2.1. Contexte

Les *archéologues* travaillent avec un programme de télédétection basé sur une technologie dite *LiDAR* (« Light Detection And Ranging ») qui permet d'acquérir des images géographiques. A partir de ces acquisitions, les archéologues génèrent plusieurs types d'images selon leurs besoins et l'un d'entre eux est l'image de relief sur laquelle les archéologues peuvent marquer manuellement les éléments archéologiques.

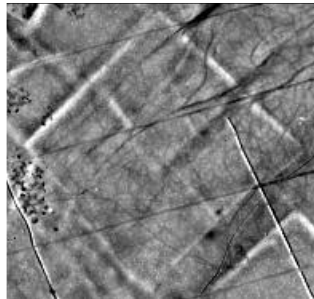


Figure 3 - Exemple d'image de relief prise à partir d'image LiDAR

Les archéologues souhaitent réaliser un *logiciel* qui permettrait d'automatiser la *détection d'éléments* archéologiques dans des images de relief. Cette solution doit permettre de générer un tableau de probabilité et non une décision binaire, de plus, il doit être open-source et doit pouvoir être utilisé *facilement sur n'importe quelle base de données*.

C'est donc dans ce contexte que la MOA propose un projet afin de trouver une solution pour prédire les probabilités pour chaque pixel de l'image son appartenance à chaque classe de relief (talus, fossé, drain, chemin bordé...).

1.2.2. Description du problème

A l'aide la technologie LiDAR, il est possible de détecter des microreliefs sous couvert forestier révélant des structures archéologiques ou naturelles invisibles à l'œil nu.

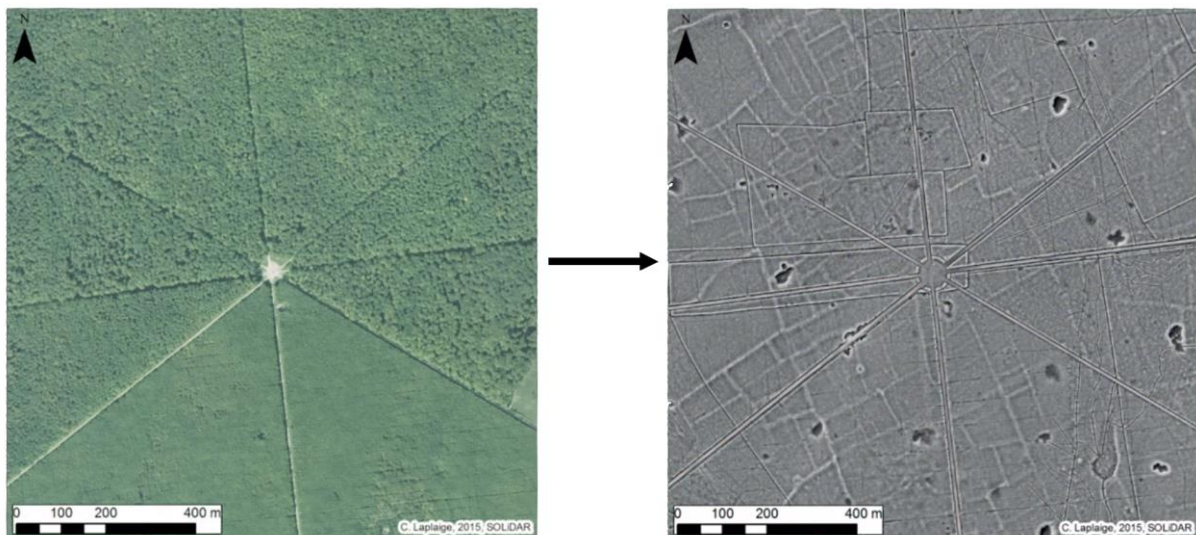


Figure 4 - Comparaison vue aérienne et acquisition par télédétection LiDAR

Les archéologues doivent ensuite manuellement les relever, les analyser puis les caractériser suivant différents types de relief. Le problème consiste à automatiser ce processus de reconnaissance.

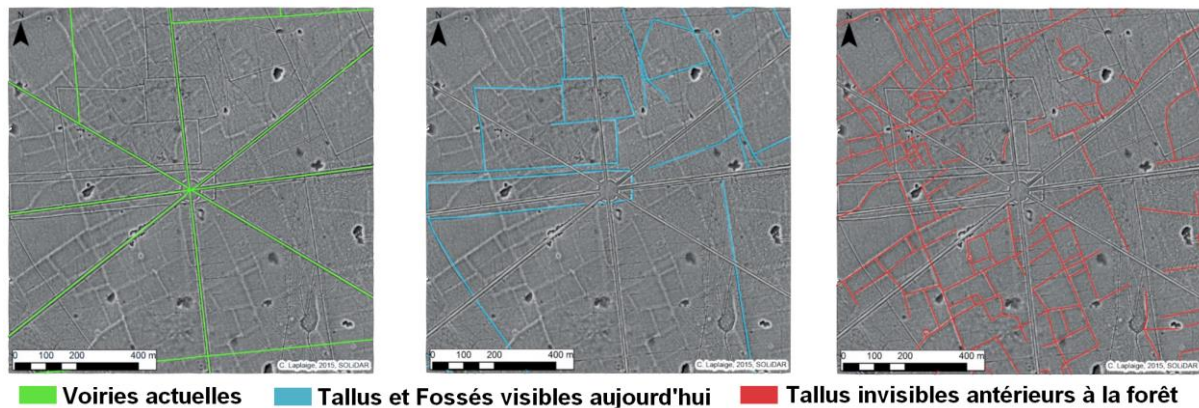


Figure 5 - Reconnaissance des différents reliefs

1.2.3. Existant

a) Logiciel

Le laboratoire possède actuellement un logiciel, répondant à ce problème, implémenté en *Python* lors de PR&D d'années antérieures. Ce dernier fonctionne correctement mais peut être amélioré au travers de nombreux domaines. Pour plus d'informations sur les différentes fonctionnalités existantes, voir Section 3.

b) Méthode de résolution

Afin de reconnaître les reliefs à partir d'une image, l'outil passe par une phase d'apprentissage avant de pouvoir classifier de nouvelles données. Pour cela, il analyse des images fournies par l'utilisateur afin que le modèle puisse « apprendre » à reconnaître ces reliefs.

La méthode d'apprentissage utilisé est un apprentissage de renforcement à l'aide de *Séparateurs à Vaste Marge (SVM)*. Pour plus d'information, veuillez consulter le chapitre « Etat de l'art ».

Le modèle peut ensuite prédire l'appartenance de chaque pixel de l'image à une classe de relief parmi celles qu'il aura appris à reconnaître.

1.2.4. Objectifs

Ce projet pouvant être qualifié de *rétro-ingénierie* a pour objectif de reprendre l'existant et de l'améliorer. Cette amélioration passe par ces points suivants :

- Revoir l'interface homme-machine afin de la rendre plus ergonomique.
- Revoir la conception du projet afin de rendre l'ajout de fonctionnalité le plus simple possible.
- Générer de la documentation interne et externe au code afin de faciliter la reprise du code pour un développeur externe au projet
- Revoir l'implémentation des algorithmes utilisés afin d'améliorer les temps d'exécution du logiciel.

2. Description générale

2.1. Environnement du projet

Ce a pour but d'être utilisé sur des machines *Windows 7* minimum ayant installé des versions ultérieures au *Python 3.5* et *OpenCV 3.1.0*.

2.2. Caractéristiques des utilisateurs

Le produit final a pour but d'être utilisé par des archéologues. Nous devons donc prendre en compte les requis suivants pour que les futurs utilisateurs puissent se servir de cette solution :

- Connaissances basiques de l'informatique ;
- Utilisateurs réguliers ;
- Connaître le fonctionnement de ou des algorithmes de résolution afin de pouvoir les paramétrer.

Ce dernier point permet de mettre en exergue la possibilité d'avoir deux types d'utilisateurs, un ayant les connaissances requises pour modifier les paramètres de résolutions et un utilisateur régulier ne se contentant qu'à utiliser les fonctionnalités principales de la solution sans se soucier de son paramétrage.

Il sera donc primordial que l'IHM permette à ce deuxième type d'utilisateurs d'être guidé tout au long de son utilisation.

2.3. Fonctionnalités du système

Les différents cas d'utilisation du système sont explicités dans le diagramme ci-dessous. La fonctionnalité principale de l'outil est bien sûr de pouvoir classer une image afin de visualiser les différents reliefs que le modèle a reconnu.

Cependant, le modèle doit avoir appris au préalable à différencier les différentes classes de reliefs. L'utilisateur doit donc pouvoir lancer un apprentissage ainsi que de tester son modèle pour vérifier que ce dernier est suffisamment compétent pour reconnaître efficacement les reliefs de l'image. Enfin, l'utilisateur doit pouvoir configurer son classificateur s'il souhaite ajouter un type de relief à reconnaître par exemple.

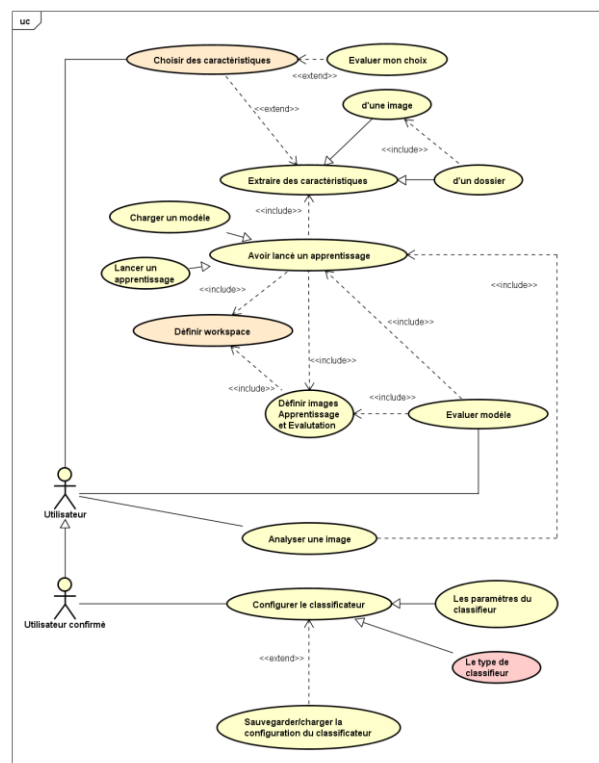


Figure 6 - Diagramme de cas d'utilisation

2.4. Structure générale du système

Le système se décompose en 4 parties :

- L'interface Homme/Machine
- Extraction des caractéristiques, pour relever différentes mesures sur chaque pixel d'une image

- Image, qui symbolise une image dans notre modèle
- Apprentissage de la machine, qui utilisera donc ces composants ci-dessus pour reconnaître les reliefs d'une image.

Pour chaque image d'entraînement, la MOA devra fournir n+1 images sources au format tif n étant le nombre de classes reconnu par le système. Ce nombre est actuellement fixé à 5. Il faudra donc une image issue de la technologie LiDAR (niveaux de gris) stocké dans le sous dossier ./train-image/img et une image binaire par classe dans les dossiers ./train-image/CX. Ces images contiendront un pixel noir si ce pixel appartient à la classe correspondant à son emplacement.

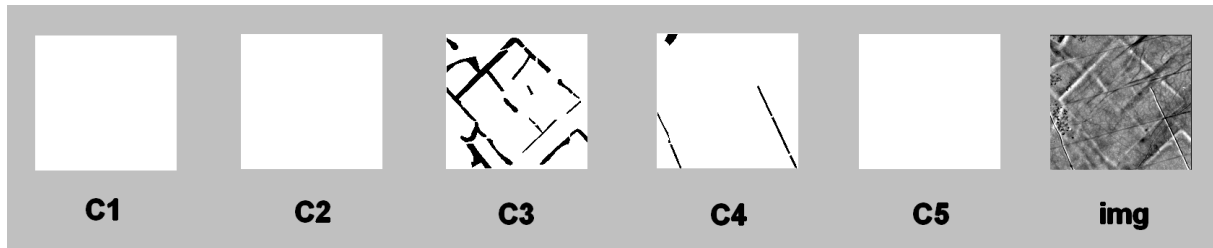


Figure 7 - Exemple des images sources à fournir par la MOA

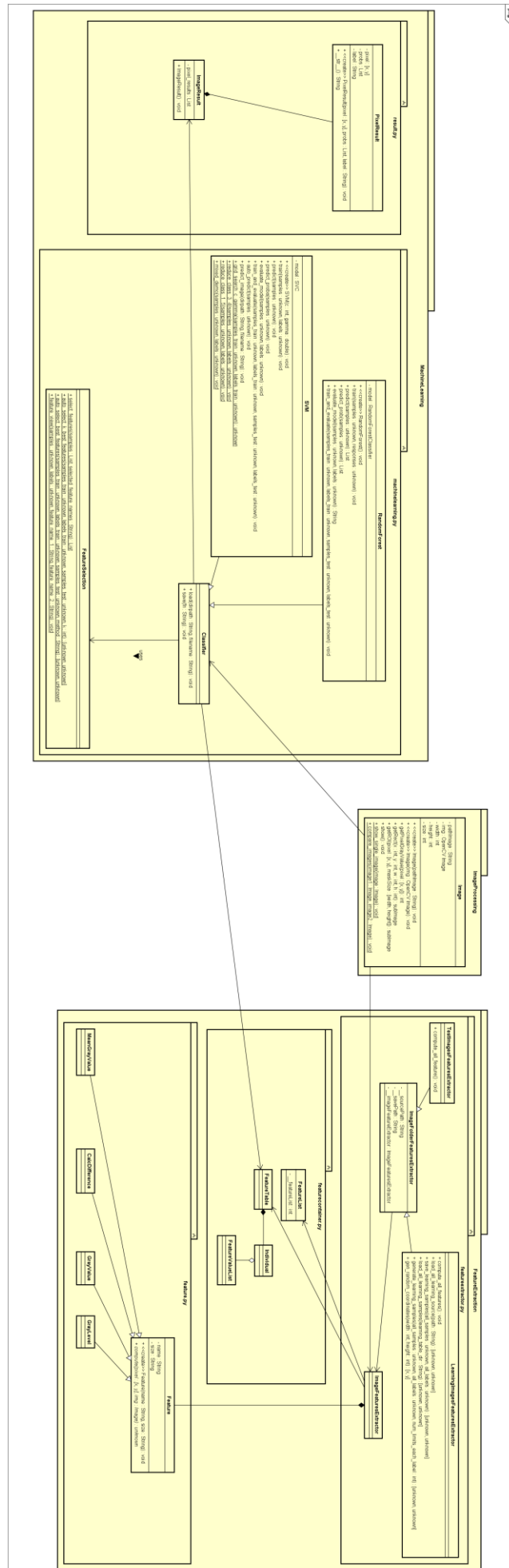


Figure 8 - Diagramme de classes de l'existant ^[1]

2.5. Description des Interfaces homme/machine

L'IHM actuelle n'est composé que d'une seule vue et ne répond donc pas très bien aux différents critères suivants :

- Pas de gestions des messages d'erreur ;
- Problèmes d'intuitivité pour l'utilisateur ;

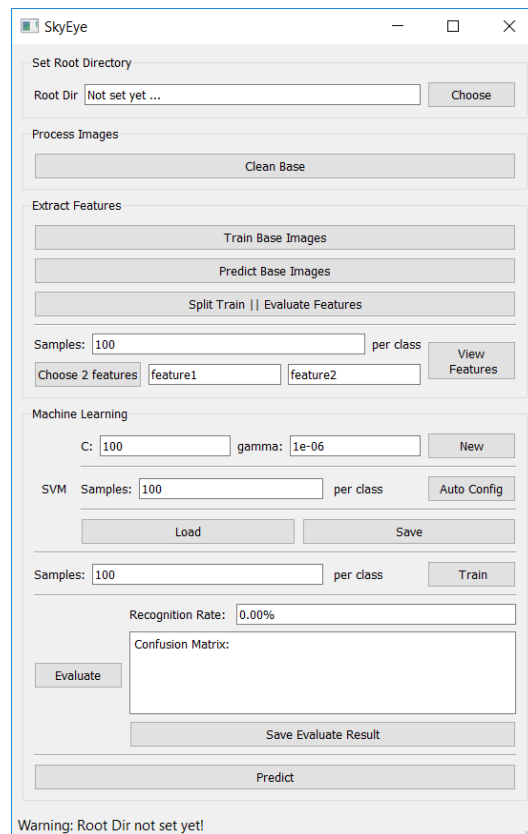


Figure 9 - IHM de l'existant

3. Spécifications fonctionnelles

3.1. Gestion du workspace

3.1.1. Description

L'utilisateur doit définir l'emplacement du dossier dans lequel il va travailler. Une fois choisi, le logiciel vérifie les dossiers dont il aura besoin pour fonctionner. Voici l'arborescence requise :

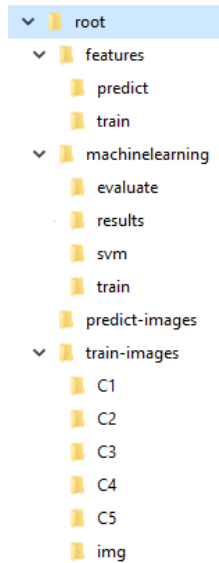


Figure 10 - Arborescence requise du workspace

Si certains dossiers sont manquants dans le workspace défini par l'utilisateur, le logiciel doit les créer afin d'obtenir l'arborescence requise.

De plus, le logiciel devra retrouver dans le dossier ./train-images le nombre de classes à reconnaître ainsi que leur noms respectifs.

Ensuite, le logiciel doit vérifier que les données sources sont conformes à la structure défini précédemment partie 2.4. Si un élément est manquant, l'utilisateur devra en être informé afin qu'il puisse remédier au problème. Si les données sont conformes, le workspace pourra alors stocker la liste des noms des images d'apprentissage.

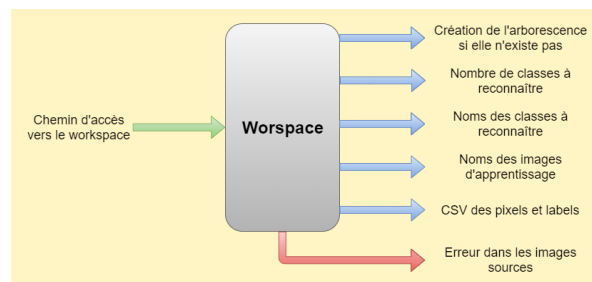


Figure 11 - Récapitulatif des fonctionnalités du workspace

Enfin, le logiciel pourra générer un fichier csv contenant les informations suivantes de chaque pixel de chaque images d'apprentissage ayant une classification défini par l'expert (le pixel est noir dans l'une des images binaires) :

- Les coordonnées x et y du pixel
- Le nom de fichier de l'image
- Le label correspondant au nom de la classe.

Evidemment, le CSV pourra être sauvegardé puis chargé afin de sauter cette étape lors d'une utilisation ultérieure. Voici un exemple de CSV de sortie :

	A	B	C	D	E
1	x	y	Nom de l'image	label	
2		0	0 img_train_1	fossé	
3		16	14 img_train_1	talus	
4		16	27 img_train_1	route	
5		19	87 img_train_1	route	
6		26	69 img_train_1	talus	
7		42	61 img_train_1	fossé	
8		47	49 img_train_1	talus	
9		54	8 img_train_1	fossé	
10		63	38 img_train_1	route	
11		68	31 img_train_1	fossé	
12		72	1 img_train_1	route	
13		79	29 img_train_1	route	
14		83	91 img_train_1	fossé	
15		87	15 img_train_1	route	
16		89	42 img_train_1	talus	
17		90	6 img_train_1	talus	
18		1	68 img_train_2	fossé	
19		17	65 img_train_2	talus	
20		20	20 img_train_2	talus	
21		35	30 img_train_2	fossé	
22		40	46 img_train_2	route	
23		44	86 img_train_2	fossé	
24		51	34 img_train_2	talus	
25		54	78 img_train_2	talus	
26		56	71 img_train_2	fossé	
27		56	88 img_train_2	route	
28		56	83 img_train_2	fossé	

Figure 12 - Exemple de CSV de sortie du workspace

3.1.2. Comparaison avec l'existant

Ces différentes fonctionnalités sont déjà implémentées, mais le concept d'objet workspace n'existe pas. Il y aura donc un travail d'analyse et de conception afin de transférer les fonctionnalités existantes dans ce nouvel objet workspace.

De plus, le nombre et les noms des classes étaient jusqu'à présent fixent, il s'agit donc d'une nouvelle fonctionnalité.

3.2. Gestion de l'extraction des caractéristiques des images d'apprentissage

3.2.1. Description

L'utilisateur doit pouvoir extraire toutes les caractéristiques de chaque image sous forme d'un fichier csv stocké dans .\features\train. Ce fichier contient des mesures pour chaque pixel de l'image comme son niveau de gris, la moyenne de gris de son voisinage, sa différence de gris avec son voisinage...

A partir de la liste de toutes les caractéristiques calculables et de la liste des images d'entraînements, et du CSV élaboré ci-dessus, l'extracteur ajoutera une colonne pour chaque caractéristique et calculera la valeur de celui-ci pour chacun des pixels.

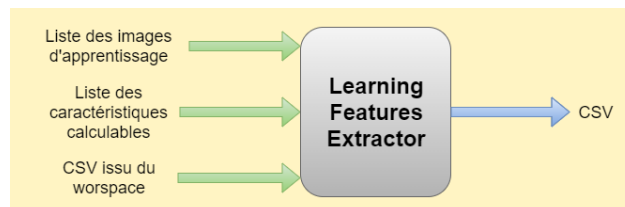


Figure 13 - Récapitulatif des fonctionnalités de l'extracteur de caractéristique des images d'apprentissage

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
26	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
28	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
29	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
31	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
32	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
33	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
34	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
35	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
36	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
37	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
38	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
39	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 14 - Exemple de fichier csv d'extraction de caractéristiques d'images d'apprentissage (base d'apprentissage)

3.2.2. Comparaison avec l'existant

Entièrement implémenté et aucune amélioration n'est prévu.

3.3. Gestion de l'extraction de caractéristiques

3.3.1. Description

L'extraction de caractéristiques va, contrairement à l'extracteur vu précédemment, n'extraire qu'un certains de caractéristiques (défini par l'utilisateur) d'une seule image (que nous souhaiterons analyser plus tard).

Un fichier CSV sera alors généré dans le dossier .\features\predict. Ce document contiendra donc une ligne par pixel de l'image et son nom de fichier sera le même que celui de l'image analysée.

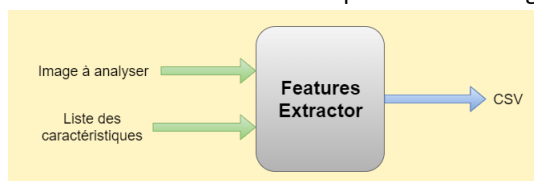


Figure 15 - Récapitulatif des fonctionnalités de l'extracteur de caractéristiques

	A	B	C	D	E
1	x	y	Gray	Avg3x3	Avg5x5
2		1	1 6.00	76.22	113.80
3		1	2 13.00	86.78	111.60
4		1	3 75.00	101.56	112.48
5		1	4 160.00	122.44	113.80
6		1	5 72.00	92.44	110.88
7		1	6 101.00	97.67	110.72
8		1	7 129.00	111.44	113.12
9		1	8 170.00	128.78	118.32
10		1	9 198.00	150.11	123.12
11		1	10 204.00	164.44	125.35
12		1	11 78.00	98.33	113.08
13		1	12 122.00	109.11	114.64
14		1	13 153.00	116.89	115.84
15		1	14 163.00	122.11	116.00
16		1	15 172.00	125.44	112.60
17		1	16 161.00	123.89	110.48
18		1	17 139.00	127.78	110.04
19		1	18 180.00	136.22	112.64
20		1	19 194.00	152.78	118.72
21		1	20 192.00	163.67	128.76
22		1	21 184.00	165.00	136.44
23		1	22 155.00	155.89	141.50
24		1	23 63.00	82.67	98.84
25		1	24 96.00	89.89	99.84
26		1	25 125.00	101.78	101.52
27		1	26 159.00	112.00	102.68

Figure 16 - Exemple de fichier csv d'extraction de caractéristiques

3.3.2. Comparaison avec l'existant

Seul le choix des caractéristiques à calculer est absent dans l'existant. Jusqu'à présent, toutes les caractéristiques sont calculées.

3.4. Gestion des classifieurs

3.4.1. Description

Le classifieur se divise en deux grandes fonctionnalités : l'apprentissage et l'analyse. Dans la première, il va exploiter notre base d'apprentissage afin de créer son modèle. Afin de reconnaître équitablement chaque classe, le classifieur n'utilise qu'un nombre X de lignes aléatoires dans la base d'apprentissage. Ce modèle peut être sauvegardé puis chargé ultérieurement afin de sauter cette étape lors des prochaines utilisations de ce logiciel.

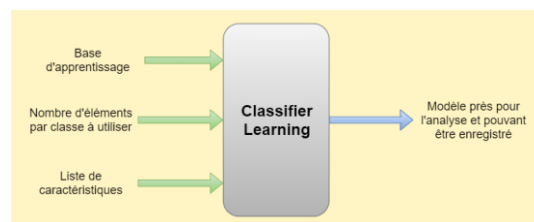


Figure 17 - Récapitulatif de l'apprentissage du classifieur

Une fois cela fait, il pourra l'exploiter pour générer, à partir du CSV d'extraction d'une image, générer un CSV contenant la probabilité de chaque pixel d'appartenir à chaque classe d'appartenance. Il sera enregistré dans le même répertoire que les résultats (voir ci-après).

	A	B	C	D	E	F	G	H
1	x	y	C0	C1	C2	C3	C4	class
2	0	0	0.26583	0.02466	0.23883	0.43434	0.03634	C3
3	0	1	0.15991	0.01844	0.27531	0.52241	0.02393	C3
4	0	2	0.23610	0.02019	0.34606	0.37673	0.02091	C3
5	0	3	0.22019	0.02449	0.46047	0.28230	0.01255	C2
6	0	4	0.19565	0.02327	0.42382	0.34626	0.01099	C2
7	0	5	0.19151	0.02773	0.67120	0.10452	0.00505	C2
8	0	6	0.19518	0.02770	0.69204	0.08006	0.00502	C2
9	0	7	0.17224	0.02508	0.68593	0.11055	0.00621	C2
10	0	8	0.14635	0.02850	0.75927	0.06130	0.00457	C2
11	0	9	0.11842	0.03066	0.76941	0.07624	0.00527	C2
12	0	10	0.20349	0.12304	0.50435	0.12007	0.04906	C2
13	0	11	0.22193	0.12280	0.51120	0.08236	0.05171	C2
14	0	12	0.26037	0.11723	0.49057	0.05966	0.07217	C2
15	0	13	0.25371	0.10353	0.53941	0.04804	0.05531	C2
16	0	14	0.35818	0.09722	0.46139	0.05135	0.03186	C2
17	0	15	0.52193	0.06790	0.35445	0.04480	0.01092	C0
18	0	16	0.50309	0.04299	0.42120	0.02547	0.00725	C0
19	0	17	0.77165	0.03387	0.16294	0.02391	0.00763	C0
20	0	18	0.74000	0.02370	0.21893	0.01243	0.00495	C0
21	0	19	0.88107	0.01882	0.08649	0.00996	0.00366	C0
22	0	20	0.90543	0.01696	0.05998	0.01470	0.00294	C0
23	0	21	0.93848	0.01320	0.03184	0.01405	0.00242	C0
24	0	22	0.95962	0.00742	0.02124	0.01041	0.00130	C0
25	0	23	0.97335	0.00750	0.00982	0.00853	0.00080	C0
26	0	24	0.98650	0.00502	0.00479	0.00329	0.00040	C0
27	0	25	0.99414	0.00311	0.00124	0.00129	0.00022	C0

Figure 18 - Exemple de résultat d'une analyse par le classifieur

3.4.2. Comparaison avec l'existant

Comme précédemment, le choix des caractéristiques à utiliser pour faire son apprentissage est une fonctionnalité pas encore implémentée.

3.5. Exploitation des résultats

3.5.1. Description

A partir du CSV de résultats vu ci-dessus, un dossier contenant les résultats sera créé dans le sous-répertoire ./machinelearning/results et sera nommé par le même nom que l'image source. Il contiendra une image tif par classe où chaque pixel labélisé à cette classe sera noir (les autres blancs). De plus, une image retravaillée de la source sera enregistré au format tif également.



Figure 20 - Images d'appartenance pour chaque classe

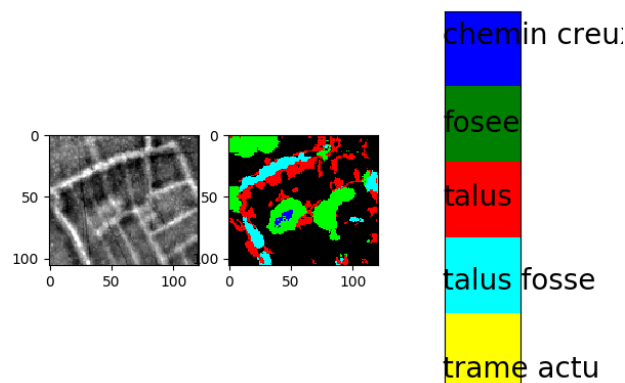


Figure 19 - Image finale générée

3.5.2. Comparaison avec l'existant

Bien qu'il soit intéressant d'avoir les images binaires afin de pouvoir faire de notre image une image source d'apprentissage. La MOA a exprimé son souhait d'obtenir des images en niveaux de gris afin qu'elle puisse visuellement avoir un esprit critique sur les résultats.

De plus, l'image final a besoin d'être retravaillé d'un point de vue esthétique.

4. Spécifications non fonctionnelles

4.1. Contraintes de développement et conception

Ce projet devra être codé en Python à partir de l'existant.

4.2. Performances

Le temps d'exécution n'est pas un enjeu important dans ce projet pour l'instant. Il devra être possible à l'avenir de l'utiliser sur des images pouvant aller jusqu'au Giga octet. Pour ce type d'image, le temps d'exécution ne devra pas dépasser la durée d'une nuit.

5. Autres tâches importantes

5.1. Génération de documentations

Dans le but de faciliter la reprise de ce projet pour les futurs développeurs, il est primordial de générer une documentation détaillée ainsi qu'un guide d'installation.

L'utilisation de Sphinx permettra de générer cette documentation à partir des commentaires laissés dans le code.

5.2. Modification de l'IHM

Il a été mis en évidence tout au long de ce document la nécessité de modifier l'interface homme-machine afin de rendre cette dernière plus intuitive pour un nouvel utilisateur.

Une esquisse devra être validé par la MOA dans un premier temps avant de l'implémenter. Elle sera élaborée en collaboration avec la MOA et la MOE afin de répondre à leurs attentes. L'utilisation du logiciel en ligne Draw IO nous permettra de travailler cette esquisse ensemble à distance.

6. Conclusion

Voir partie Développement du rapport.

BIBLIOGRAPHIE

- [1] : Bai, S. (2016-2017). *Reconnaissance automatisée d'éléments archéologiques dans des images LiDAR*. Tours: Ecole Polytechnique de l'Université François Rabelais de Tours.



Webographie

- [1] Microsoft AZURE. *Comment choisir les algorithmes dans Microsoft Azure Machine Learning*. [En ligne ; Page disponible le 25-avril-2017]. 2017. URL : <https://docs.microsoft.com/fr-fr/azure/machine-learning/studio/algorithm-choice>.
- [2] Charles CROUSPEYRE. *Comment les Réseaux de neurones à convolution fonctionnent*. 2017. URL : <https://medium.com/@CharlesCrouspeyre/comment-les-r%C3%A9seaux-de-neurones-%C3%A0-convolution-fonctionnent-b288519dbcf8>.
- [3] LIBERTOX. *Les drogues et le cerveau*. 2012. URL : http://www.libertox.com/?page_id=125.
- [4] WIKIPÉDIA. *Intelligence artificielle — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 17-novembre-2017]. 2017. URL : http://fr.wikipedia.org/w/index.php?title=Intelligence_artificielle&oldid=142675431.
- [5] WIKIPÉDIA. *Machine à vecteurs de support — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 26-juillet-2017]. 2017. URL : http://fr.wikipedia.org/w/index.php?title=Machine_%C3%A0_vecteurs_de_support&oldid=139259844.
- [6] WIKIPÉDIA. *Réseau neuronal convolutif — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 2-décembre-2017]. 2017. URL : http://fr.wikipedia.org/w/index.php?title=R%C3%A9seau_neuronal_convolutif&oldid=143133982.



Bibliographie

- [1] Shuo BAI. « Reconnaissance automatisée d'éléments archéologiques dans des images Li-DAR ». Projet Recherche & Développement. Ecole Polytechnique de l'Université François Rabelais de Tours, 2017.
- [2] Hubert CARDOT. *Cours RF*. Présentation. Ecole Polytechnique de l'Université François Rabelais de Tours, 2017.
- [3] Michael A. NIELSEN. *Neural Networks and Deep Learning*. Sous la dir. de Determination PRESS. 2015. URL : <http://neuralnetworksanddeeplearning.com/index.html>.

Plateforme d'extraction et de caractérisation automatique d'éléments d'intérêt dans les images archéologiques par apprentissage interactif

Ronan GUILLAUME

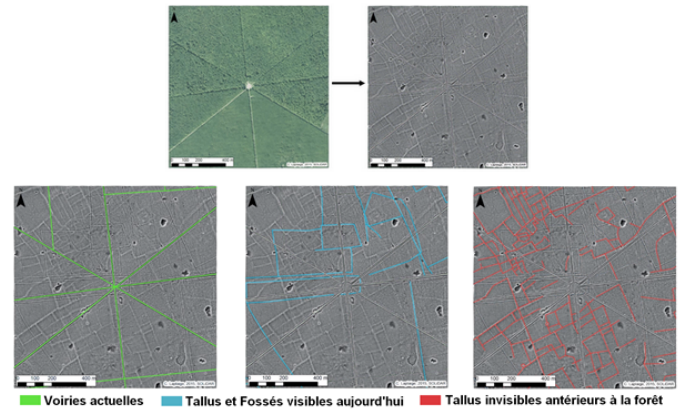
Encadrement : Jean-Yves Ramel

Contexte

Reprise d'un projet en collaboration avec des chercheurs en archéologie dans le cadre du projet SOLiDAR.

Mise en place d'un système de détection automatique de vestiges archéologiques à partir d'images LiDAR.

Utilisation de techniques d'apprentissage automatiques

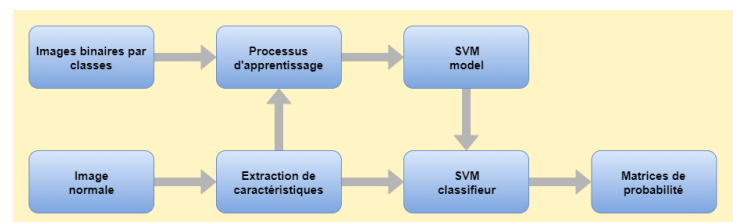


Exemple du processus effectué manuellement

Améliorations

Tâche à remplir :

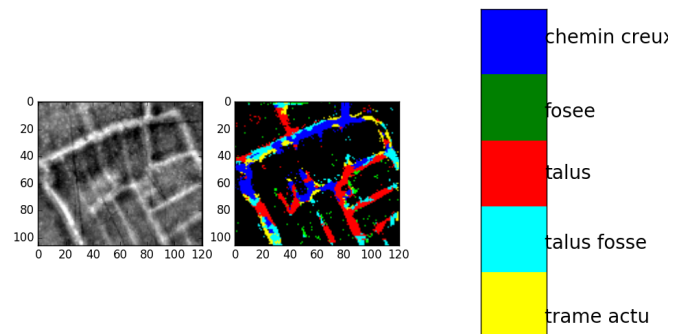
- Création d'une nouvelle IHM plus intuitive pour l'utilisateur.
- Gestion dynamique des classes de formes à reconnaître par le système.
- Gestion du modèle d'apprentissage.
- Pouvoir paramétrer son système.



Fonctionnement du système

Objectifs

- Améliorer le taux de reconnaissance du système en donnant une large gamme de paramétrisation pour l'utilisateur.
- Faciliter l'utilisation de l'application.
- Pouvoir reconnaître tout types de formes



Résultat visuel de l'existant

Plateforme d'extraction et de caractérisation automatique d'éléments d'intérêt dans les images archéologiques par apprentissage interactif

Ronan GUILLAUME

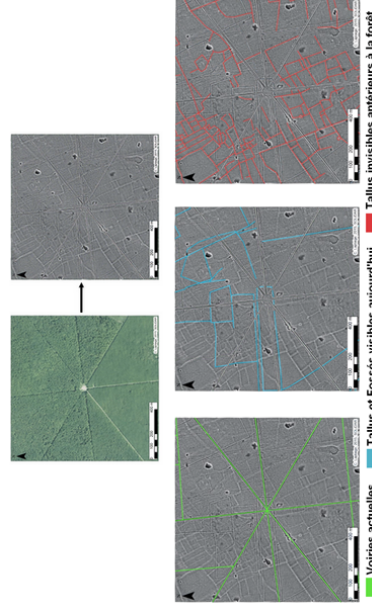
Encadrement : Jean-Yves Ramel

Contexte

Reprise d'un projet en collaboration avec des chercheurs en archéologie dans le cadre du projet SOLiDAR.

Mise en place d'un système de détection automatique de vestiges archéologiques à partir d'images LiDAR.

Utilisation de techniques d'apprentissage automatiques



Exemple du processus effectué manuellement

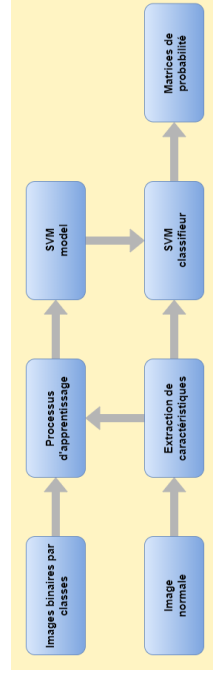
Améliorations

Tâche à remplir :

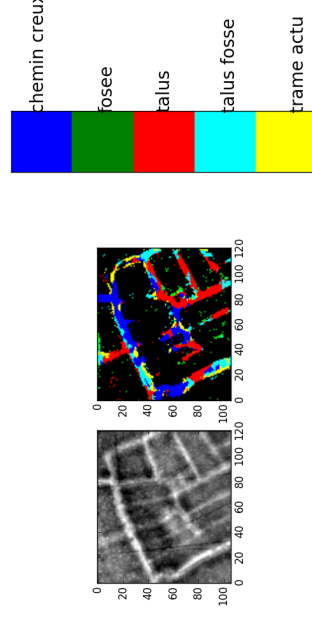
- Création d'une nouvelle IHM plus intuitive pour l'utilisateur.
- Gestion dynamique des classes de formes à reconnaître par le système.
- Gestion du modèle d'apprentissage.
- Pouvoir paramétrer son système.

Objectifs

- Améliorer le taux de reconnaissance du système en donnant une large gamme de paramétrisation pour l'utilisateur.
- Faciliter l'utilisation de l'application.
- Pouvoir reconnaître tout types de formes



Fonctionnement du système



Résultat visuel de l'existant



Plateforme d'extraction et de caractérisation automatique d'éléments d'intérêt dans les images archéologiques par apprentissage interactif

Résumé

Ce rapport décrit le projet recherche & développement. L'objectif du projet est de détecter les vestiges archéologiques à partir d'images dérivées de données LiDAR. En phase de recherche, ce rapport décrit une nouvelle méthode d'apprentissage permettant au système d'être plus performant. En phase de développement, il faudra améliorer l'existant en lui ajoutant un workspace, le choix des caractéristiques à exploiter ainsi qu'une nouvelle IHM.

Mots-clés

Reconnaissance de formes; Apprentissage; Classifieur; Caractéristique

Abstract

This report describes the research & development project. The objective of the project is to detect archaeological remains on images derived from LiDAR data. In the research phase, this report describes a new learning method to create a more effective system. In the development phase, I will improve the existing by adding a workspace, the choice of features to exploit and a new IHM.

Keywords

Pattern Recognition; Learning; Classifier; Feature