ENSG Géomatique
ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

Ecole Nationale des Sciences Géographiques

UNIVERSITÉ PARIS-EST

Université Paris-Est

-PROJET INFORMATIQUE -
Rapport d'analyse

Mastère spécialisé ® Photogrammétrie, Positionnement, Mesure de Déformations

Développement d'une interface graphique pour améliorer des classifications de manière interactive

ENSG Géomatique
ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

CHUPIN Clémence

Novembre-Décembre 2017

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

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

Table des matières

Glossaire et sigles utiles

ENSG École Nationale des Sciences Géographiques
PPMD Mastère Spécialsé Photogrammétrie, Positionnement et Mesure de Déformations
IGN Institut National de l'Information Géographique et Forestière
LaSTIG Laboratoire en Sciences et Technologies de l'Information Géographique
MATIS Méthodes d'Analyses pour le Traitement d'Images et la Stéréorestitution
LiDAR Light Detection And Ranging
MNT Modèle Numérique de Terrain
MNS Modèle Numérique de Surface
LoD Level Of Details
SIG Système d'Information Géographique
SHP Shapefile
CSV Comma Separated Value
TIFF Tagged Image File Format

Introduction

Dans le cadre des projets informatiques des étudiants du Mastère Spécialisé PPMD de l'ENSG, le laboratoire MATIS de l'IGN a proposé un sujet se rattachant à la problématique des modèles 3D urbains. Ce sujet s'inscrit dans le cadre de la thèse de M.ENNAFII Oussama, intitulée *"Evaluation and selection of 3D city modelling techniques"*. Pour détecter et caractériser les erreurs des maquettes 3D urbaines, une méthode d'auto-qualification par classification supervisée a été mise en place. Pour permettre la transition vers une classification active,le projet propose de mettre en place une interface graphique pour interagir avec un utilisateur.

La réalisation de ce projet se divise en deux grandes étapes. La première partie est une phase d'analyse du sujet et de rencontres avec les commanditaires. Elle a pour objectif de détailler les enjeux du projet, et de réfléchir aux méthodes d'implémentation à utiliser. La seconde phase correspond à la programmation du logiciel. A l'issu de cette étape, le projet doit être opérationnel, même si l'ensemble des fonctionnalités prévues n'est pas implémentée. De plus, une documentation doit permettre de détailler les différentes fonctionnalités du programme.

Dans ce premier rapport, on s'attardera donc à détailler les étapes d'analyse du sujet et les problématiques qui lui sont liées. Dans un premier temps, on cherchera à définir précisément le contexte d'étude, les besoins des commanditaires et les principaux objectifs du projet. Cela permettra, dans un second temps, de détailler la structure de l'application. On analysera les données nécessaires et leurs contraintes, ainsi que les fonctionnalités à implémenter et leur organisation. Enfin, une dernière partie permettra de justifier les choix techniques réalisés pour l'implémentation.

Chapitre 1
Définition du sujet

Le présent chapitre a pour objectif de définir le sujet : cela permettra de préciser le contexte de l'étude et de redéfinir la problématique, avant d'analyser les enjeux du projet.

1.1 Contexte du sujet

Grâce au développement des techniques LiDAR et aux prises de vues aériennes et satellites, il est possible de représenter le terrain (création de Modèles Numériques de Terrain). Afin d'avoir une bonne maitrise du milieu urbain, le processus de reconstruction 3D permet d'avoir une modélisation plus avancée. Pour construire ces maquettes 3D urbaines, on utilise un nuage de points, créé à partir de données LiDAR ou par reconstruction photogrammétrique.

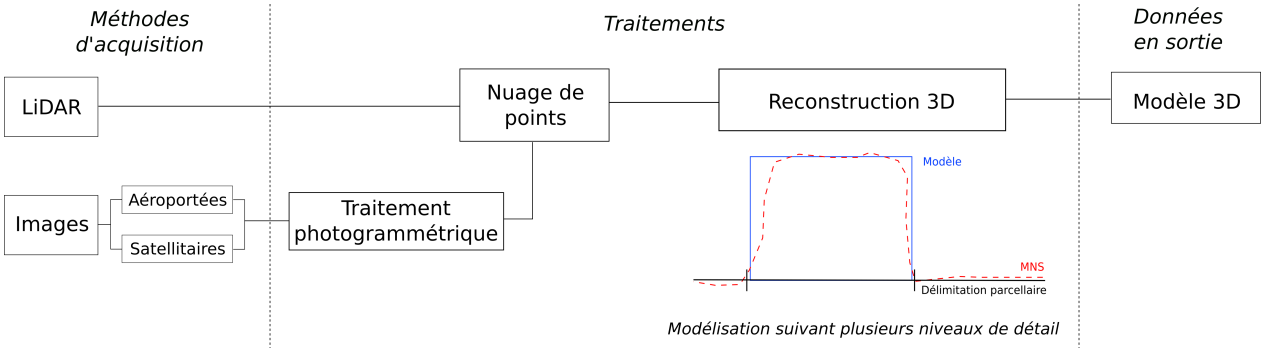

Figure 1.1 – Chaîne simplifiée de reconstruction 3D urbaine

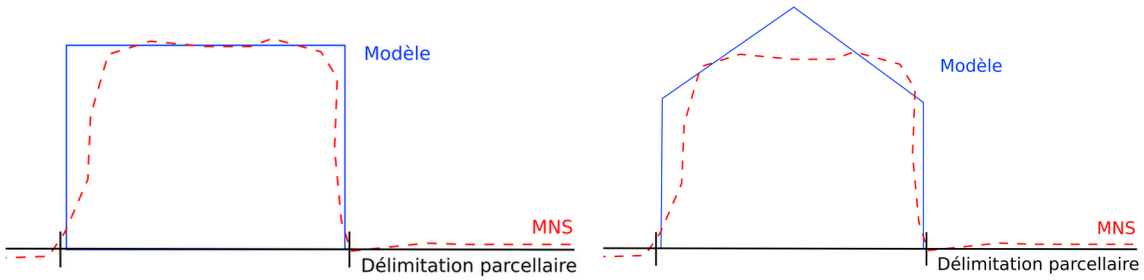

Figure 1.2 – Principe de la reconstruction urbaine

Le niveau de détail de la maquette (LoD) peut varier selon les objectifs de l'utilisateur. Différentes modélisations sont possibles :

→ Manhattan (LoD1 = toits plats)

→ Biplan (LoD2 = toits pentus)

→ Modélisation précise (LoD3 = représentation des microstructures - cheminées, gouttières, ...)

Plus le modèle est évolué, plus le niveau de détail est important mais plus la généralisation est compliquée.

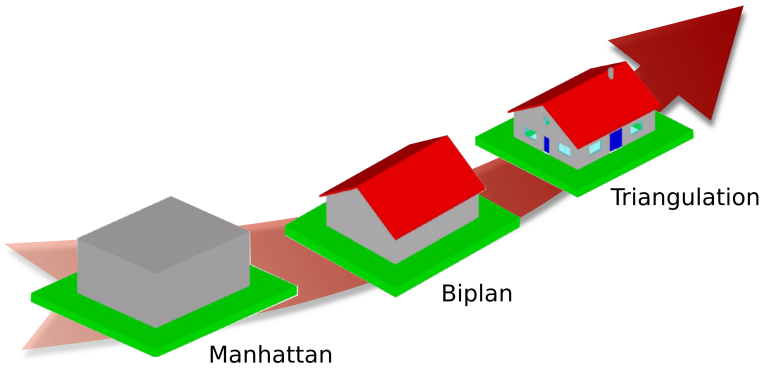

FIGURE 1.3 – Impact du niveau de détail sur la modélisation [1]

Aujourd'hui, les techniques de reconstruction automatiques ne sont pas opérationnelles. L'utilisateur doit intervenir pour corriger manuellement les erreurs. Le passage à l'échelle de la chaîne de production devient donc problématique. Afin d'alléger la phase de correction, on cherche à automatiser la qualification de ces modèles. Pour cela, deux méthodes sont envisageables : la qualification avec des données de référence (nécessite un modèle de référence), et l'auto-qualification.

En matière de reconstruction 3D, le laboratoire MATIS de l'IGN a mis au point la méthode de reconstruction automatique BATI-3D. Le projet présenté ici s'inscrit dans le cadre des recherches du laboratoire pour la reconstruction 3D des bâtiments à partir de Modèles Numériques de Surface (MNS). L'objectif est d'automatiser la correction des maquettes 3D urbaines. Cela implique d'automatiser en premier la phase d'auto-qualification des erreurs.

1.2 Problématique

Pour détecter et caractériser les erreurs de reconstruction, l'équipe de recherche a mis en place une méthode d'auto-qualification. Ce processus passe par la réalisation d'une classification supervisée des entités reconstruites, afin de leur associer une classe d'erreur.

Les différents types d'erreur sont ordonnés hiérarchiquement selon des classes, choisies en fonction du niveau de détail de la reconstruction. Ainsi, on peut définir une typologie des erreurs rencontrées (cf. annexe A).

L'objectif du projet est d'évoluer vers une classification active. En effet, l'intervention d'un utilisateur permettrait de valider (ou non) les résultats de la classification, afin d'affiner le modèle de prédiction entrainé.

1. Sources : OGC CityGML Implementation Specification 1 (Research Center Karlsruhe)

Le projet de développement présenté dans ce rapport propose donc de mettre en place un outil graphique d'aide à la classification active. L'objectif est de réaliser une interface permettant à un utilisateur de valider un résultat de classification, ou dans le cas contraire de renseigner la bonne erreur. Pour cela, les objets d'intérêt (orthophoto et bâtiments orthoprojetés) doivent pouvoir être visualisés selon une hiérarchie à définir.

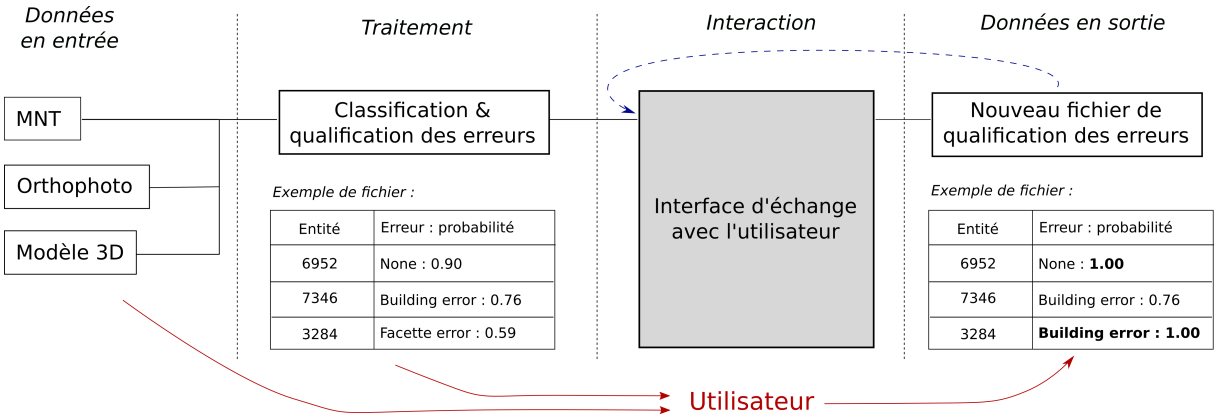

FIGURE 1.4 – Chaîne de traitement simplifiée du projet

1.3 Analyse des besoins

Après avoir défini précisément le contexte du projet et sa problématique, il est possible d'analyser ses principaux objectifs. Ainsi, les enjeux du projet sont liés à l'affichage et à la validation des résultats de la classification.

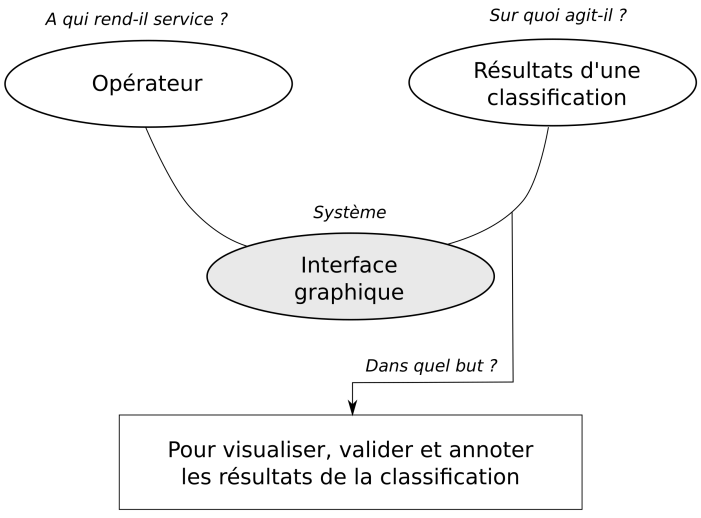

FIGURE 1.5 – Objectifs généraux du projet

Dans un premier temps,ce projet s'adresse aux utilisateurs du processus de qualification des données de reconstruction 3D urbaine. Cependant, l'interface réalisée peut être amenée à évoluer et à servir pour visualiser d'autres cas de classification.

Chapitre 2
Analyse fonctionnelle

Le présent chapitre a pour objectif de présenter la structuration de l'application. La modélisation du problème va influencer l'organisation des données du programme. De plus, différentes fonctionnalités doivent être implémentées pour agir sur ces données.

2.1 Contraintes de modélisation du projet

Les principales contraintes de développement de l'interface sont liées à la modélisation du problème. Deux choix sont possibles : une modélisation multi-classe, et une modélisation multi-label. Dans le premier cas, les objets classés sont répartis sur plusieurs classes, tout en sachant qu'un objet ne peut appartenir qu'à une seule classe. Dans le cas de la modélisation multi-label, un objet peut appartenir à plusieurs labels, chacun proposant un choix binaire (appartenance/non-appartenance).

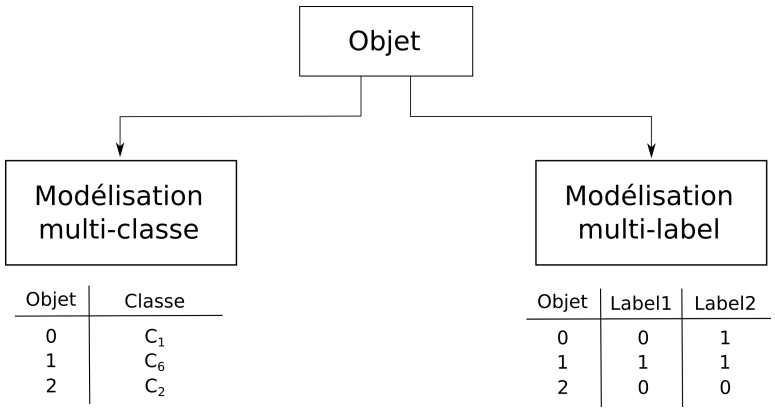

Figure 2.1 – Modélisation du problème

Le choix de la modélisation du problème a un impact sur le formalisme des données en entrée et en sortie du programme. Le choix de l'interface à montrer sera aussi déterminé par la modélisation. Ainsi, dans le cas multi-classe, un objet classifié ne sera lié qu'à une seule erreur. Cela implique que l'utilisateur ne peut choisir qu'une seule classe d'erreur.

Pour l'implémentation de l'interface, le modèle multi-classe a été privilégié pour premier objectif. Une fois cet objectif atteint, on pourra évoluer vers une modélisation multi-label. L'interface programmée doit donc être flexible pour prendre en compte les différents formalismes.

Le choix de modélisation du problème impacte le formalisme des données nécessaires. Ainsi, après avoir décrit les contraintes liées au projet, on peut analyser les données en entrée et en sortie du programme.

2.2 Organisation des données

2.2.1 Données en entrée

La première étape d'analyse concerne les données en entrée, dans le cas d'une modélisation multi-classe. On considère 4 jeux de données en entrée :

→ Un fichier CSV (Comma-Separated Values) qui regroupe les classes d'erreurs possibles et leur description.

→ Un fichier CSV qui regroupe les résultats de la phase de classification. Pour chaque entité, on retrouve son identifiant, sa classe et sa probabilité d'appartenance à cette classe.

→ Un dossier qui regroupe les emprises des entités au format shapefile (SHP). L'identifiant de l'entité se retrouve dans le nom du fichier d'emprise (*ex : « 2516.SHP » pour l'entité 2516*).

→ Une orthoimage au format TIFF de la zone analysée, spatialement superposable aux fichiers d'emprise.

L'ensemble de ces données doit être fourni par l'utilisateur avant l'exécution du programme. Pour cela, une fenêtre de chargement des fichiers doit être implémentée.

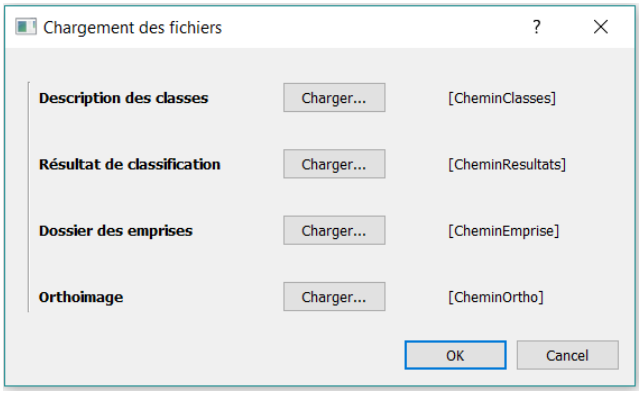

FIGURE 2.2 – Maquette de l'interface de chargement des fichiers

2.2.2 Données en sortie

En sortie de l'interface, un seul fichier est retourné :

→ Un fichier CSV qui regroupe les résultats de l'interaction avec l'utilisateur. On retrouve un formalisme identique au fichier des résultats de classification donné en entrée (identifiant, classe, probabilité). Deux méthodes sont envisageables : une première hypothèse consiste à sortir un tableau avec l'ensemble des entités testées (qu'elles soient modifiées ou non), et une seconde à ne restituer que les entités qui ont été modifiées.

2.2.3 Structuration des données

Lecture des fichiers CSV

Les données au format CSV (Comma-Separated Values) concernent les fichiers qui regroupent les classes et les résultats de classification. Ce format permet d'enregistrer des données tabulaires sous forme de valeurs séparées par des virgules.

Dans le programme, les fichiers seront traduits de manières différentes :

→ Pour le fichier qui regroupe les classes, les données seront enregistrées dans une variable de type dictionnaire. Ces données ne vont pas évoluer au cours du processus, ce qui justifie ce formalisme. Il sera ainsi possible d'accéder aux descriptions des classes via les valeurs des clés du dictionnaire.

→ Pour le fichier qui contient les résultats de classification, les données seront enregistrées comme une liste de tuples. En effet, ce type de variable est facilement manipulable avec Python. Cette liste servira pour filtrer les bâtiments à afficher.

Lecture des données graphiques

Le format SHP est un format de fichier utilisé pour les systèmes d'informations géographiques (SIG). Il permet de stocker des entités SIG vectorielles. Dans notre cas, on utilise ce format pour les emprises des bâtiments orthoprojetés. Après avoir sélectionné les entités à présenter, on utilisera ces données de géométrie pour créer une classe "Bâtiment".

La lecture de l'orthophoto au format TIFF (Tagged Image File Format) permet d'obtenir des informations quant à la géométrie de l'image (coordonnée du point origine, taille du pixel,...). Elle ne sera lu que lors de l'affichage pour limiter l'enregistrement de données.

Pour le projet, on considère que les emprises et l'orthophoto sont représentées dans le même repère. Il ne sera donc pas nécessaire de gérer les projections.

2.3 Analyse fonctionnelle

Après avoir analysé les données nécessaires au programme et leur formalisme, on peut définir les différentes fonctionnalités du projet et leur organisation.

2.3.1 Principales fonctionnalités du programme

L'interface doit permettre de visualiser et contrôler les résultats d'une classification.

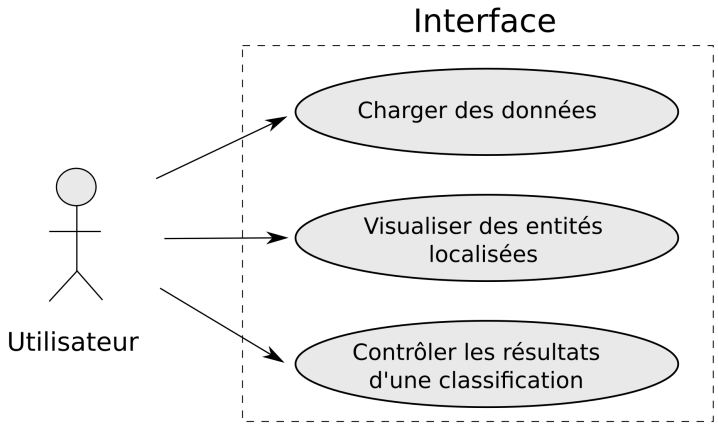

FIGURE 2.3 – Cas d'utilisation

2.3.2 Organisation générale des fonctions

Le diagramme fonctionnel ci-dessous permet de représenter sous forme de blocs fonctionnels l'ensemble du système étudié. Ces fonctionnalités doivent permettre de répondre aux cas d'utilisation de l'interface.

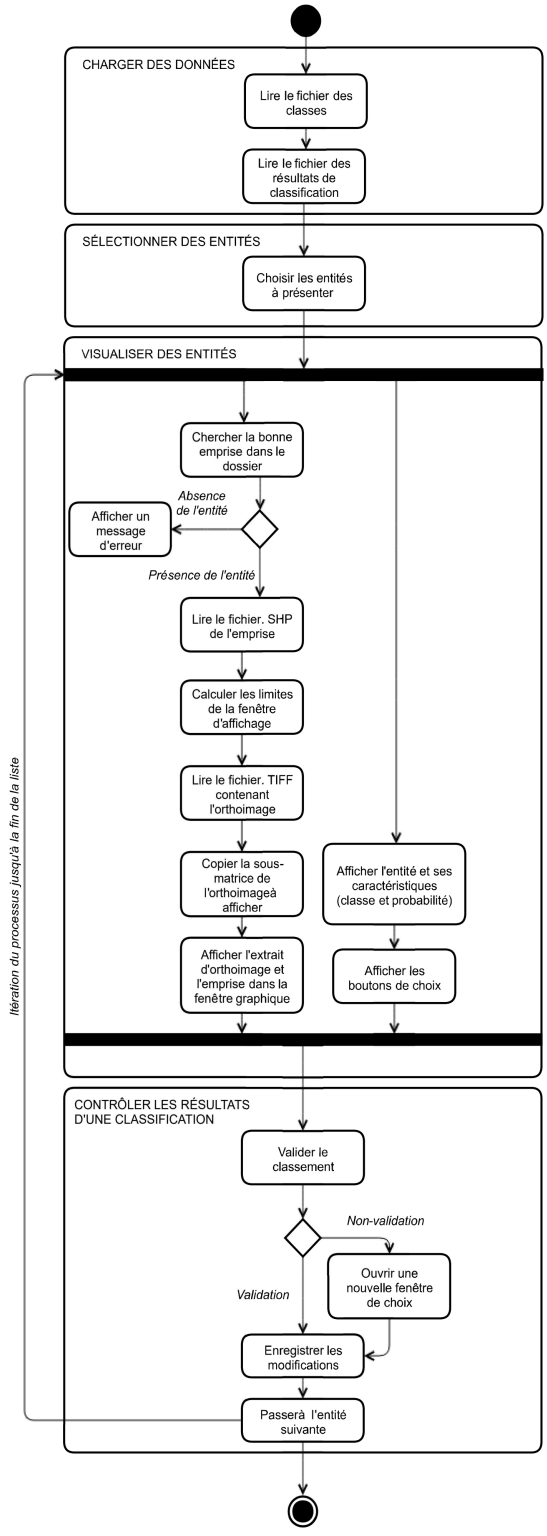

FIGURE 2.4 – Diagramme fonctionnel

2.3.3 Détail des fonctions

Après avoir schématisé l'ensemble des fonctionnalités à implémenter, on détaille le fonctionnement des principales fonctions.

Fonction de sélection des entités

Cette fonction doit permettre à l'utilisateur de choisir la stratégie d'affichage des entités résultant de la classification. En effet, l'objectif n'est pas que l'utilisateur contrôle l'ensemble des données, mais seulement celles qui influent le plus sur la classification.

Plusieurs stratégies d'affichages peuvent être envisagées :

1. L'affichage de toutes les entités
2. L'affichage d'entités aléatoirement choisies
3. L'affichage des entités dont la probabilité d'appartenance est la plus faible
4. L'affichage des entités ayant des conflits de classification (cas des erreurs ayant des probabilités très proches)

Dans les deux derniers cas, des entités aléatoires devront également être affichées. En effet, si l'on souhaite améliorer le processus global de classification, l'utilisateur doit agir sur toutes les classes.

Deux types de traitements sont envisageables :

1. Un traitement offline, qui sélectionne les entités à afficher indépendamment de la requalification par l'utilisateur
2. Un traitement online, qui va sélectionner un échantillon de données qui va être remis à jour selon les résultats du reclassement

Dans un premier temps, on se limitera au filtrage offline afin de faciliter l'implémentation et de se concentrer sur le fonctionnement global de l'interface. De plus, les différentes méthodes de filtrage seront organisées sous forme de classes pour faciliter la transition de l'une à l'autre.

Manipulation des fichiers SHP des bâtiments orthoprojetés

Une fois les entités à afficher sélectionnées, on cherche à lire les fichiers SHP correspondant. Ces emprises sont stockées dans un dossier unique, et leur nom correspond à l'identifiant de l'entité. Dans la première implémentation, en cas d'absence de l'emprise dans le dossier, cette fonction devra afficher une fenêtre d'erreur à l'utilisateur.

Grâce à la lecture des fichiers, on renseigne des objets de type "Bâtiment" avec un identifiant, une géométrie, une classe et une probabilité.

Une dernière étape consistera à calculer, pour chaque emprise, les bornes d'affichage qui lui sont liées. Cela nécessite de déterminer la fenêtre d'affichage ($[x_{min},y_{min}]$ et $[x_{max},y_{max}]$) lors de la lecture du fichier. Les marges (m_x,m_y) définies par l'utilisateur sont ensuite rajoutées aux coordonnées pour obtenir les angles repères de la fenêtre graphique ($[x_h,y_h]$ et $[x_b,y_b]$).

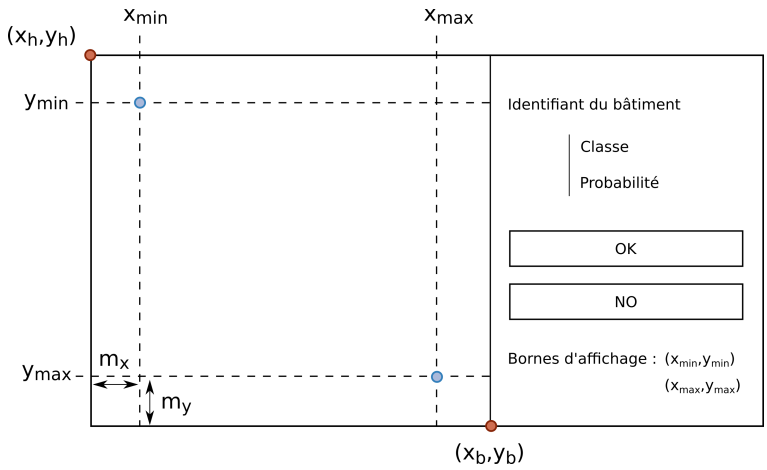

FIGURE 2.5 – Schéma d'organisation de l'interface

Analyse de l'orthophoto

La lecture du fichier TIFF de l'orthophoto doit nous permettre de trouver son point d'origine ainsi que la taille d'un pixel. Les emprises étant spatialement superposables à l'orthophoto, on utilise les valeurs des bornes d'affichage des emprises pour calculer la matrice d'orthoimage à afficher. Dans la fenêtre graphique, on lui superpose l'emprise du bâtiment.

Phase d'interaction avec l'utilisateur

L'affichage des informations relatives à l'entité permet à l'utilisateur de contrôler les résultats de la classification :

→ Si l'utilisateur valide le résultat, celui-ci est enregistré et la fenêtre affiche l'entité suivante.

→ Si l'utilisateur ne valide pas le résultat, une nouvelle fenêtre s'ouvre et propose les autres choix possibles. L'utilisateur ne peut sélectionner qu'une seule proposition, puis valide. Une fois le résultat enregistré, on affiche l'entité suivante.

2.4 Conception de l'interface

Après avoir analysé les données et les fonctionnalités de l'interface, on s'intéresse à l'organisation du programme.

2.4.1 Méthode Modèle/Vue/Contrôleur

Pour faciliter l'écriture et la compréhension du code de l'interface, on utilisera la méthode Modèle/Vue/Contrôleur :

→ Le fichier "Modèle" contiendra les données à afficher en entrée et les résultats en sortie.

→ Le fichier "Vue" contiendra le code de l'interface graphique.

→ Le fichier "Contrôleur" contiendra les définitions informatiques des actions effectuées par l'utilisateur dans l'interface (*ex : actions à exécuter lors de la validation de la classification, ...*). Ces actions viendront modifier les données contenues dans le modèle, ce qui va entrainer un changement dans la vue.

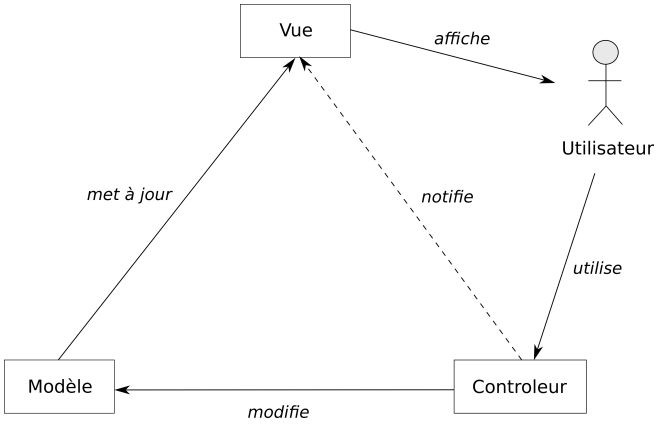

FIGURE 2.6 – Agencement Modèle/Vue/Contrôleur

2.4.2 Diagramme de classe

Les différents fichiers définis plus haut vont permettre d'agencer le code de manière claire. A l'intérieur de ces trois fichiers, on définira les différentes fonctions mais aussi des classes.

La partie "Vue" correspond à l'interface graphique. Comme on le verra par la suite, on s'appuiera sur Qt et QtCreator pour mettre au point cette fenêtre. Ainsi, les classes présentes dans le fichier vue seront gérées par Qt.

La partie "Modèle" est celle qui définit les actions réalisables par l'utilisateur. Dans ce fichier, on pense définir deux classes :

→ Une classe abstraite "Statégie" qui sélectionnera les entités à afficher en appliquant un filtre sur les données en entrée. Les types stratégies seront définies dans des classes qui hériteront des propriétés de "Stratégie".

→ Une classe "Bâtiment" qui utilisera les entités filtrées. Cette classe associera à chaque entité un identifiant, une géométrie, une classe et une probabilité.

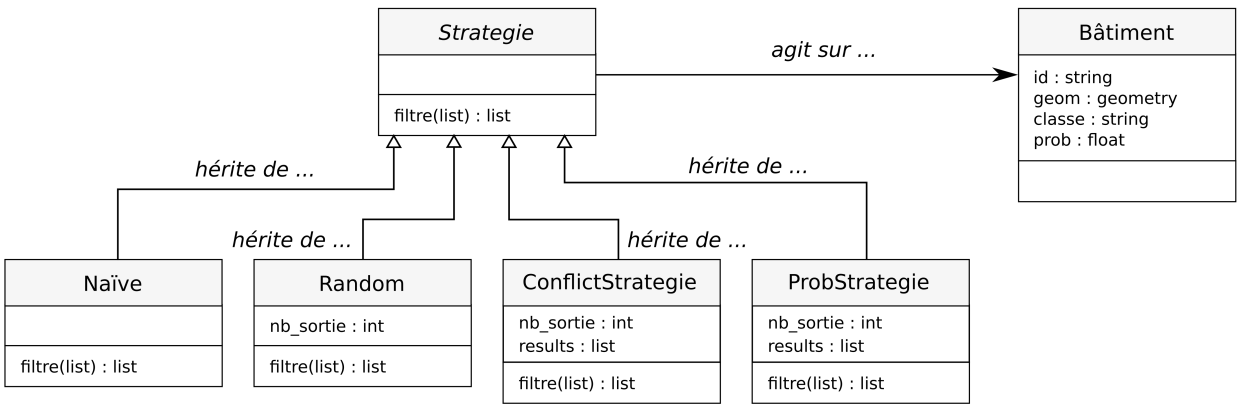

FIGURE 2.7 – Diagramme de classe

2.5 Maquette du projet

Après avoir défini l'ensemble des contraintes, des données, des fonctionnalités et de leur implémentation, on réalise une première ébauche de l'interface. La fenêtre principale permet de visualiser les entités et leur classe. On prévoit aussi une fenêtre pour charger les données, et une autre pour choisir la nouvelle classe en cas de mauvaise qualification de l'erreur.

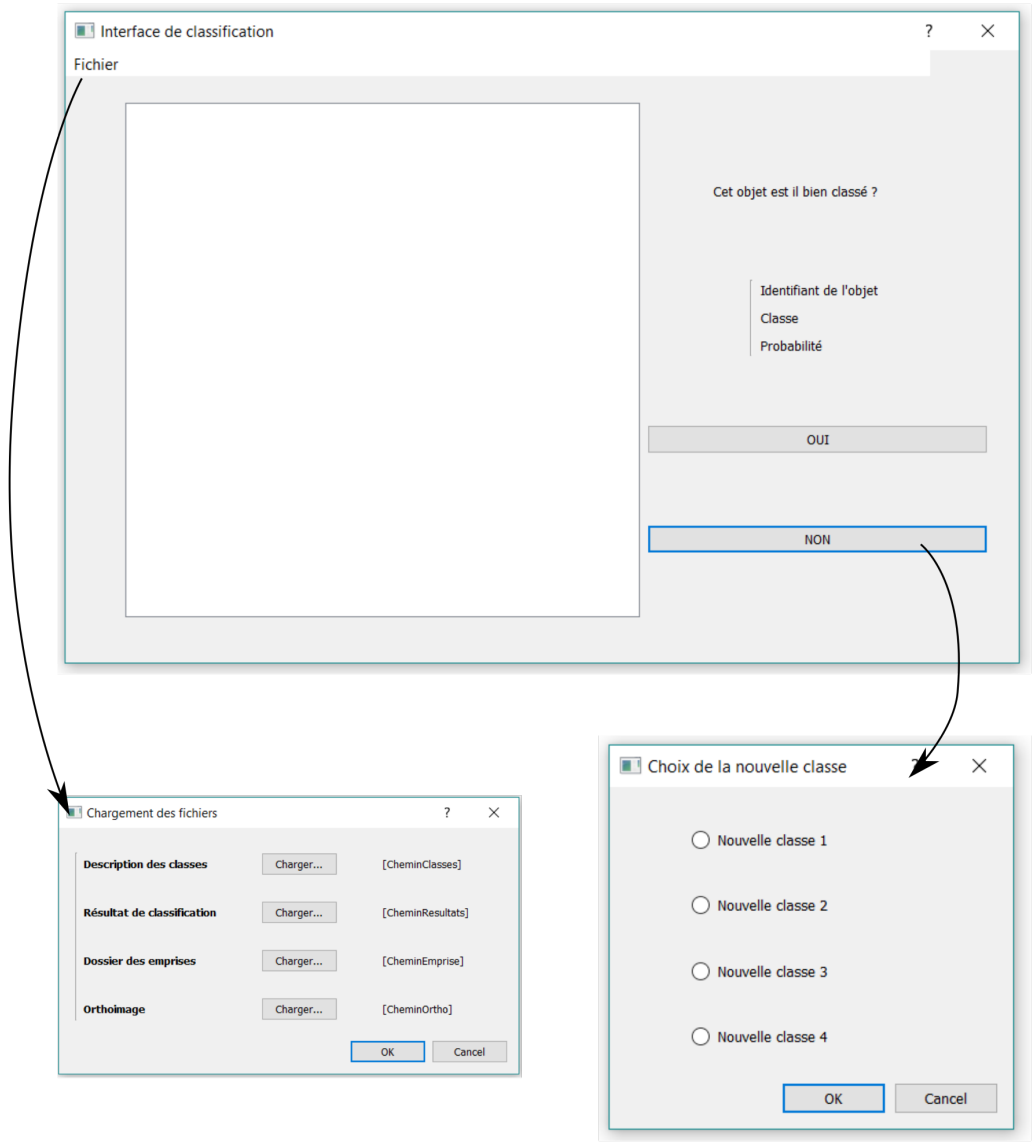

FIGURE 2.8 – Maquette de l'interface.

Les donnée et les fonctionnalités présentées dans ce chapitre s'appuient sur des choix techniques. Le prochain développement permettra de justifier ces choix.

Chapitre 3
Analyse technique

Différents choix techniques ont été effectués pour implémenter ce projet. Ce chapitre va permettre de décrire les structures existantes et de justifier les différents choix techniques réalisés.

3.1 Description de l'existant

Cette section vise à décrire les procédés existants sur lesquels s'appuie le projet.

3.1.1 Programme de classification

La contextualisation du projet a permis de montrer que, pour corriger les maquettes 3D urbaines, il est nécessaire de qualifier les erreurs de reconstruction. Cette qualification peut passer par une comparaison avec un modèle de référence. Cependant, ce processus nécessite de mettre au point un modèle précis servant de comparaison. La reconstruction automatique perd donc tout son intérêt.

Au sein du laboratoire MATIS, les recherches de M. ENNAFII Oussama ont donc pour objectif l'auto-qualification de la maquette 3D urbaine. Cette auto-qualification se traduit par une classification des erreurs de reconstruction. Ainsi, le processus n'utilise en entrée que la maquette 3D urbaine. Le résultat du traitement est un fichier CSV contenant l'identifiant des entités, leur classe d'erreur et leur probabilité.

3.1.2 Format des données

Dans les précédents chapitres, on a pu voir que différents types de données étaient utilisées :

- → Les fichiers de données textuelles sont au format CSV (Comma-Separated Values). Dans ces fichiers, les données sont séparées par des virgules. Ce type de format est assez répandu, et est largement utilisé par les logiciels tableurs. Le format CSV a été choisi pour sa lisibilité par rapport à des formats de type .TXT.
- → L'orthophoto est fournie au format TIFF (Tagged Image File Format). C'est un format de fichier courant pour les images numériques et lu par de nombreux logiciels. De plus, c'est un format assez flexible, et qui peut contenir des données sur la géométrie de l'image.
- → Les données vectorielles sont au format SHP (Shapefile). C'est un format courant pour les données SIG, qui permet d'enregistrer la géométrie d'une entité vectorielle. Dans notre cas, les données sont extraites du logiciel QGIS, qui est un logiciel SIG, ce qui justifie le format SHP.

3.2 Choix techniques

Après avoir détaillé les données existantes sur lesquelles s'appuie le projet, on peut analyser les différents choix de développement réalisés.

3.2.1 Création d'une interface indépendante de QGIS

Le projet a pour objectif de développer une interface graphique. Ainsi, au lieu de choisir de créer une nouvelle interface indépendante, la réalisation d'un plugin QGIS aurait pu être envisagée. Ce choix a été imposé par les commanditaires du projet, mais il peut être justifié.

En effet, l'objectif premier du projet est de réaliser une interface modulaire, qui ne dépend pas des évolutions et des choix techniques de QGIS. De plus, cette interface peut être amenée à être utilisée à plus grande échelle (IGN, MATIS, ...). Cela implique une certaine flexibilité dans l'implémentation et un détachement de QGIS. Enfin, l'utilisation de QGIS implique de charger à chaque fois l'ensemble de ses modules qui ne sont pas tous nécessaires au projet.

Ainsi, c'est la création d'une interface indépendante de QGIS qui a été retenue.

3.2.2 Utilisation de Qt

Pour créer des interfaces graphiques, plusieurs outils peuvent être utilisés :

- → La solution Winform et l'outil Interface Builder de l'environnement Cocoa permettent de réaliser des interfaces graphiques. Cependant, elles sont spécifiques à un système d'exploitation (respectivement Windows et Apple).
- → Si on considère l'environnement Unix, on peut également penser à l'outil GTK + (The GIMP Toolkit). La portabilité sous Windows est possible mais reste complexe à mettre en œuvre.
- → On peut également considérer l'association HTML 5 et JavaScript, qui permet de réaliser de puissantes interfaces graphiques. Cependant, ces environnements évoluent vite, ce qui pose des problèmes de maintenance de l'interface.
- → La bibliothèque graphique WxWidget permet de réaliser des interfaces portatives. Cependant, elle est aujourd'hui moins utilisée que Qt pour la création d'interface.

Ainsi, face à ces principaux arguments, la bibliothèque multi-plateforme Qt a été privilégiée pour la réalisation de cette interface.

3.2.3 Utilisation du langage Python

L'utilisation de Qt pour la réalisation de l'interface graphique ne justifie pas le choix du langage de programmation. En effet, il est possible d'utiliser Qt avec les langages C++ ou Python. De façon générale, le C++ est plus performant pour des interfaces graphiques importantes. Dans notre cas, l'interface envisagée ne nécessite pas une grande efficacité de traitement.

Ainsi, afin de faciliter le codage, l'utilisation du langage Python a été privilégiée.

L'interface graphique voulue pour ce projet utilisera donc la bibliothèque Qt associée à un script Python. Cela lui permettra d'être utilisée par tous les environnements informatiques, et elle pourra être modulée sans prendre en compte le formalisme de QGIS.

Conclusion

Cette première phase d'analyse a permis de mieux définir les enjeux liés au projet. Pour une néophyte dans le domaine de la reconstruction 3D et de la classification, cette étape était obligatoire. Elle m'a donnée le temps nécessaire pour comprendre le sujet et me l'approprier. De plus, elle a permis de développer les futures phases de l'implémentation. Même si certaines problématiques vont apparaitre au cours de la programmation, de nombreux pièges auront été évités, et le code réalisé sera plus organisé.

Dans un premier temps, la phase de programmation aura pour objectif d'implémenter une solution simple mais fonctionnelle. Si les tests réalisés durant le codage sont concluants, d'autres fonctionnalités pourront ensuite être développées. Cette phase de programmation a été organisée selon un calendrier prévisionnel, qui pourra évoluer au cours du processus.

Séance	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Finalisation de l'interface															
Chargement des données															
Sélection des données															
Affichage des données															
Phase d'interaction															
Améliorations possibles															
Rédaction des documents															

FIGURE 3.1 – Calendrier prévisionnel de programmation

Avant de clore ce rapport, je tiens à remercier les commanditaires de ce projet, M. MALLET Clément et M. ENNAFII Oussama, de me faire confiance pour la réalisation de cette interface. Je remercie plus particulièrement Oussama pour la patience dont il a fait preuve lors de cette phase d'analyse !

Table des figures

Annexes

Typologie des erreurs de classification

Annexe A

Si on analyse la figure suivante[1], on constate que le premier niveau de classement correspond à la simple dualité : erreur/ absence d'erreur.

Le second niveau permet 3 choix :

1. Erreur inqualifiable (Ex : bâtiment caché, bâtiment coupé,...)
2. Erreur concernant l'ensemble d'un bâtiment (Ex : erreur de hauteur, sous-segmentation, ...)
   → Niveau de détail : LOD 0/1
3. Erreur sur les facettes reconstruites
   → Niveau de détail : LOD 2

1. Sources : Oussama ENNAFII

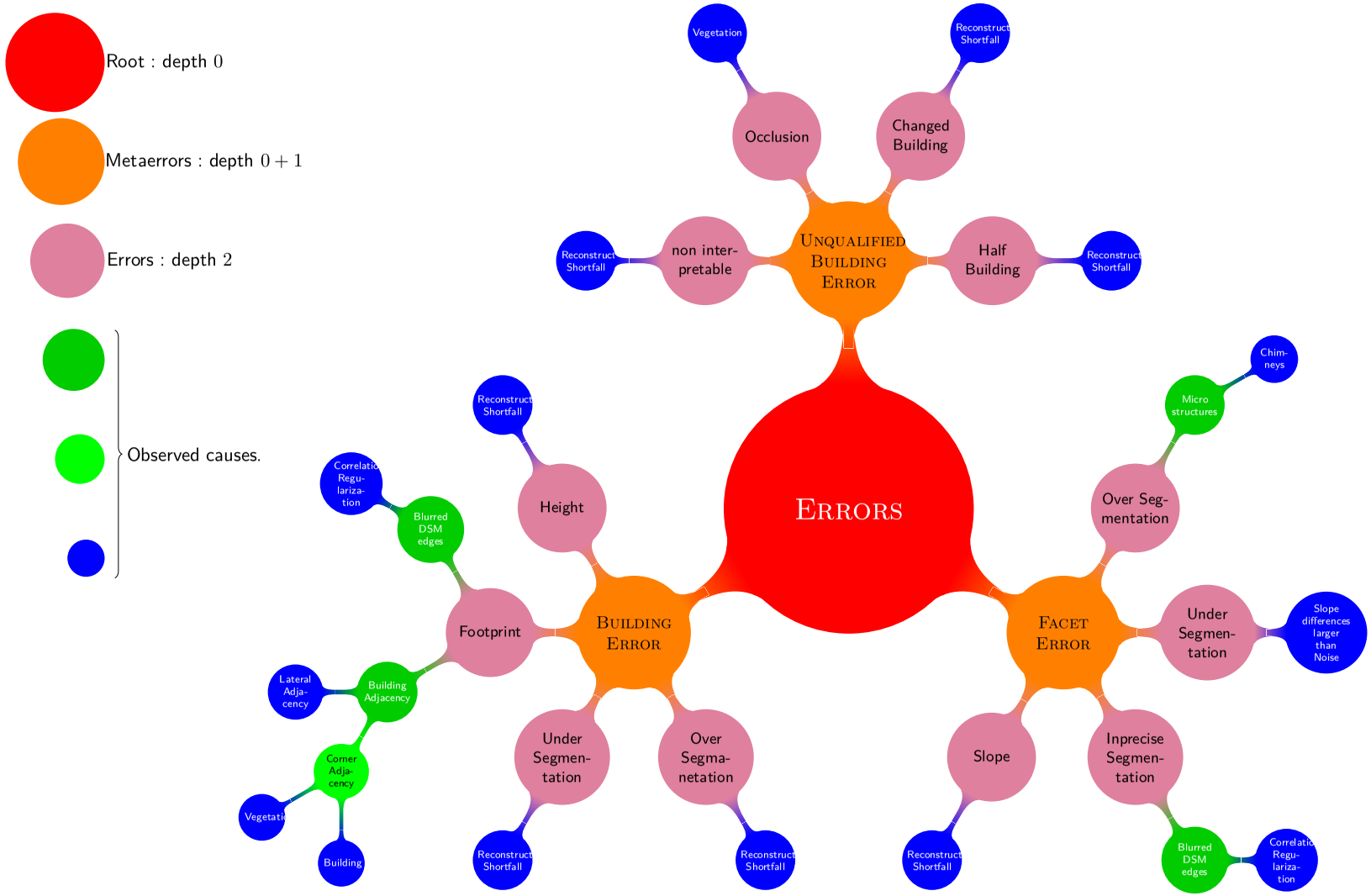

FIGURE A.1 – Typologie des erreurs de classification.