Introduction :

Ce document exhaustif offre une vue complète du projet en cours, allant des diagrammes UML à la mise en œuvre pratique du code. Il a pour objectif de fournir une référence claire et concise aux développeurs, testeurs et utilisateurs, couvrant tous les aspects essentiels du projet, des structures choisies aux algorithmes implémentés. Que vous soyez impliqué dans le développement, les tests unitaires ou simplement à la recherche d'une compréhension globale, ce document a été conçu pour être une ressource incontournable.

1. Fichiers UML :

- Cas d'utilisation
- Diagrammes de séquence associés aux cas d'utilisation
- Diagrammes de classe ou fonctions et méthodes

2. Codes :

- Code fonctionnant sur mon poste
- Code remodifié pour fonctionner sur n'importe quel poste

Dans chacun de ces 2 code l'on retrouve en début l'algorithme liée au code suivis des définitions de leurs doc string et leur doc test puis le code commentées puis en toutes fin certain test unitaire comme vue en TP ils ne seront pas forcément tous présent ou tous 100% fonctionnel.

3. Configurations :

- Code de configuration pour cx_freeze (conversion en .exe)

4. Tests Unitaires :

- Pour le code fonctionnant sur mon poste(fichier a part)

5. Fichiers Exécutables :

- .exe de l'application fonctionnant sur mon poste
- .exe de l'application fonctionnant sur n'importe quel poste

6. Algorithmes :

- Inclus dans les deux codes avec une nuance expliquée dans le code utilisant des chemins relatifs

7. Documents PDF :

- Répertoire des éléments dans le code
- Justificatif des structures choisies
- Guide d'utilisation et d'architecture
- Déroulement du projet
- Code des diagramme uml

8. Données Excel :

- Trois fichiers Excel pour sourcer et manipuler les données de l'application

Conclusion :

Ce dossier constitue une ressource complète pour le projet, présentant de manière organisée toutes les composantes nécessaires à la compréhension et à l'utilisation de l'application. Des fichiers UML aux codes sources et exécutables, en passant par des documents explicatifs, il offre une vue d'ensemble structurée. Les fichiers Excel et les algorithmes ajoutent une dimension pratique, illustrant l'interaction de l'application avec les données. En résumé, cette compilation facilite le développement et l'utilisation du projet sous tous ses aspects.

Introduction :

Le processus de développement de notre projet a été rigoureux, débutant par une phase de recherche pour définir clairement les attentes du rendu final, centré autour d'une interface homme-machine (IHM). Une fois les objectifs définis, l'équipe s'est attelée à l'identification des données nécessaires, impliquant des recherches et la constitution d'un dossier complet avec des documents et des photos, désormais regroupés dans le dossier "Photos" et docs sous format Excel.

Phase de Modélisation :

La phase suivante a consisté à modéliser le projet à l'aide de différents diagrammes tels que le diagramme de cas d'utilisation et les diagrammes de séquence associés. Ces modèles ont servi de cadre au projet, bien que certains aient été ajustés au fur et à mesure des développements et des améliorations identifiées en cours de route.

Phase de Codage :

Le développement a débuté avec la formulation de règles précises, établissant la manière dont le code devait être structuré. Face à l'absence d'idées claires sur la mise en œuvre des différentes tâches, nous avons procédé par essais et erreurs, explorant différentes techniques de filtrage et de manipulation d'images. Ce processus itératif a permis d'aboutir aux algorithmes actuels, chaque étape du code étant soigneusement élaborée, anticipant une utilisation potentielle dans d'autres projets.

Traitement des Images :

La première partie du code a été consacrée à la modification des photos pour les rendre exploitables. Des transformations de forme et de couleur, notamment en niveaux de gris, ont été appliquées pour préparer les images à la prochaine étape du traitement.

Algorithme de Détection de Forêts :

Le traitement suivant a consisté à définir les contours des forêts sur les photos. Trois filtres Canny avec des paramètres différents ont été testés, et le filtre Canny de type médian s'est avéré le plus précis dans de nombreuses situations. En remplissant ensuite les contours en blanc, nous avons obtenu une représentation visuelle de la déforestation.

Calculs et Exportation des Données :

Une échelle a été établie à partir des photos, fixée à 20 km pour 38 pixels, pour mesurer la surface de déforestation. La partie suivante du code a impliqué des calculs détaillés et l'exportation des données vers des tableaux Excel, suivie d'une mise en forme des données pour faciliter la lisibilité et l'analyse.

Interface Homme-Machine (IHM) :

La dernière phase a été consacrée à la création d'interfaces utilisateur. Initialement, l'interface a été réalisée avec Tkinter pour sa simplicité et son adéquation aux objectifs pédagogiques. Cependant, dans une optique de développement personnel, nous avons également mis en place une interface HTML, ajoutant des couches de complexité avec l'intégration de CSS et JavaScript pour améliorer l'esthétique et les fonctionnalités.

Conclusion :

Ce périple de développement a été marqué par des itérations constantes, allant de la modélisation à la mise en œuvre pratique. Chaque phase du projet a été minutieusement planifiée pour aboutir à une application complète, capable de traiter des images satellitaires et de fournir des données précieuses sur la déforestation. Les défis rencontrés ont été des opportunités d'apprentissage, contribuant à l'évolution du projet vers une solution plus aboutie.

Code pour les diagrammes

Cas d’utilisation

```
@startuml
!theme aws-orange
left to right direction
actor Utilisateur as user
rectangle "Application" {
  usecase "Importer Images" as import
  usecase "Appliquer Filtres" as filters
  usecase "Afficher Résultats" as display

  user --> import : Utilise
  user --> filters : Utilise
  user --> display : Utilise
}
@enduml
```

Diagramme de sequence

```
@startuml
!theme aws-orange
actor Utilisateur as user
participant "Application" as app
participant "Interface" as interface
database "Base de Données" as database

user -> app : Importe Images
activate app

app -> app : Applique Filtres
app -> database : Enregistre Données

app -> interface : Affiche Résultats
activate interface

interface -> user : Présente Résultats
deactivate interface

deactivate app
@enduml
```

Diagramme de classe

```
@startuml
!theme aws-orange
class ImageProcessor {
  - image: Image
  + importImages(images: List): void
  + applyFilters(): void
  + displayResults(): void
}

class Database {
  - data: List
  + saveData(data: List): void
}

class UserInterface {
  + showResults(data: List): void
}

ImageProcessor --> Database
ImageProcessor --> UserInterface
@enduml
```

Introduction :

L'application que nous présentons offre une solution robuste pour l'analyse de vues satellitaires en vue de surveiller la déforestation. Ce document explicatif vise à fournir une compréhension approfondie de l'architecture et du fonctionnement visuel de l'application. Nous détaillerons le processus depuis l'importation des images jusqu'à l'affichage des résultats sur une interface utilisateur intuitive.

Fonctionnement et Architecture :

Le processus débute par l'ajout d'images de vues satellitaires dans le dossier dédié. Bien que toutes les images ne soient pas garanties de fonctionner de manière optimale, certaines images d'exemple sont fournies dans ce dossier. À l'intérieur du dossier de seuillage, déjà prérempli dans notre cas, un sous-dossier est créé pour chaque image initiale. Chaque sous-dossier contient une version de l'image en niveaux de gris, une version isolant la couleur verte sur fond noir, ainsi que trois images avec des applications de l'algorithme de Canny à des niveaux de précision différents. Enfin, une image avec un remplissage en blanc du résultat optimal de l'algorithme de Canny est conservée.

Le code, après les diverses manipulations d'image, alimente un fichier Excel avec des données pertinentes, notamment la surface des pixels blancs, convertie en une mesure de surface en fonction de l'échelle déterminée sur les images. Ce processus se base sur une échelle de 20 km pour 38 pixels, ce qui semble cohérent avec nos recherches. Les données sont ensuite renvoyées dans l'Excel, suivi de la création d'une interface utilisateur tkinter. À la fermeture de la fenêtre tkinter, l'interface HTML s'ouvre, offrant une expérience intuitive. Un simple clic sur "Commencer" permet de choisir une essence d'arbres dans la liste déroulante, affichant des données similaires à l'interface tkinter à gauche et des images défilantes à droite. Les chemins d'accès aux bases de données Excel sont également accessibles en haut à droite.

Cependant, il est important de noter que la partie HTML de l'interface n'est pas dynamique dans cette version. Les chemins d'accès aux données et aux images ne sont pas mis à jour automatiquement, nécessitant une intervention manuelle. Cette limitation sera prise en compte dans les développements futurs de l'application pour offrir une expérience encore plus fluide.

Le html et le java script étant du bonus.

Conclusion :

Ce document offre une vue détaillée du fonctionnement de l'application, de l'importation des images à la visualisation des résultats. Bien que des éléments manuels subsistent, tels que la mise à jour des données et des chemins d'images, l'interface globale offre une expérience utilisateur cohérente. L'application se positionne comme un outil potentiellement précieux pour surveiller la déforestation et analyser les vues satellitaires de manière efficace.

Voici une liste des fonctions présentes dans Le code, ainsi qu'une brève description de leur action :

1. modifier_fichier

   - Arguments : `input_path` (chemin du fichier d'entrée), `output_path` (chemin du fichier de sortie)

   - Action : Lit le contenu du fichier d'entrée, modifie chaque ligne pour ajouter une balise `<img>` autour du contenu, puis écrit le résultat dans le fichier de sortie.

2. trouver_chemins_images

   - Arguments : `dossier` (chemin du dossier à explorer), `extensions` (tuple des extensions d'images à rechercher, par défaut ('jpg', 'jpeg', 'png', 'gif'))

   - Action : Parcourt récursivement le dossier spécifié pour trouver les chemins des fichiers d'images avec les extensions spécifiées.

3. enregistrer_chemins_dans_fichier

   - Arguments : `chemins_images` (liste des chemins des fichiers d'images), `fichier_sortie` (chemin du fichier de sortie, par défaut 'chemins_images.txt')

   - Action : Enregistre les chemins des fichiers d'images dans un fichier texte.

4. ouvrir_html

   - Arguments : `chemin_fichier` (chemin du fichier HTML à ouvrir)

   - Action : Ouvre le fichier HTML dans le navigateur par défaut.

5. process_images_in_folder

   - Arguments : `input_folder` (chemin du dossier d'entrée), `output_folder` (chemin du dossier de sortie), `lower_green` (plage inférieure de la couleur verte en format HSV), `upper_green` (plage supérieure de la couleur verte en format HSV)

   - Action : Traite les images dans le dossier d'entrée en appliquant différents filtres et enregistrant les résultats dans le dossier de sortie.

6. detecter_contours_et_remplir

   - Arguments : `image_path` (chemin de l'image à traiter)

   - Action : Charge une image, détecte les contours blancs, remplit les zones délimitées par les contours, puis enregistre l'image résultante.

7. compter_pixels_et_calculer_superficie

- Arguments : `image` (tableau Numpy représentant une image)

- Action : Compte le nombre total de pixels, le nombre de pixels blancs, puis calcule la superficie en km^2 à partir des pixels blancs.

8. supprimer_extension

- Arguments : `nom_fichier` (nom du fichier avec ou sans extension)

- Action : Supprime l'extension du nom du fichier et retourne le résultat.

9. main

- Arguments : `nom_fichier` (chemin du fichier Excel à traiter)

- Action : Ouvre le fichier Excel, remplace la cellule A1 par "Image", parcourt les cellules de la colonne 1 et supprime l'extension du contenu, puis sauvegarde le fichier Excel modifié.

10. afficher_resultats

- Action : Affiche les résultats dans une zone de texte en fonction de la sélection d'une essence d'arbre dans une liste.

Ces descriptions fournissent une vue d'ensemble des actions réalisées par chaque fonction contennue dans le code.

Voici une liste des choix structurels présentes dans Le code, ainsi qu'une brève justification :

1. Listes :

- utilisez des listes pour stocker les chemins des images (`chemins_images`), les noms uniques d'essences d'arbres (`essences_uniques`), et pour afficher les résultats dans l'interface utilisateur (`essences_uniques`).

- Justification : Les listes sont adaptées lorsque a une collection ordonnée d'éléments. Elles offrent une flexibilité pour ajouter, supprimer ou accéder aux éléments par index.

2. Dictionnaires :

- utilisez un dictionnaire pour spécifier les entêtes dans le classeur Excel (`{"Nom de l'image", "Proportion de blanc (%)", "Surface en $km^2$"}`).

- Justification : Les dictionnaires sont utiles pour stocker des paires clé-valeur. Dans ce cas, ils fournissent une manière lisible d'associer des entêtes aux colonnes du classeur Excel.

3. DataFrame Pandas :

- utilisez des DataFrames Pandas pour manipuler et analyser des données tabulaires provenant des fichiers Excel (`arbres_df`, `surfaces_df`, `result_df`).

- Justification : Les DataFrames Pandas sont bien adaptés pour travailler avec des données tabulaires. Ils offrent des fonctionnalités puissantes pour filtrer, fusionner, et analyser les données, ce qui facilite le traitement et la manipulation des données.

4. Numpy Array :

- utilisez des tableaux Numpy pour effectuer des opérations mathématiques sur des images, par exemple, dans la fonction `compter_pixels_et_calculer_superficie`.

- Justification : Les tableaux Numpy sont optimisés pour les opérations vectorielles et matricielles, ce qui les rend adaptés pour le traitement d'images et le calcul numérique.

5. Classe Workbook d'Openpyxl :

- utilisez la classe `Workbook` d'Openpyxl pour créer et manipuler des fichiers Excel.

- Justification : La classe Workbook d'Openpyxl est spécialement conçue pour travailler avec des fichiers Excel en Python. Elle offre des méthodes pour créer, lire et écrire des feuilles de calcul Excel.Et comme je vous le disais dans un cadre d'appréhension et en corélation cela m'entraine à manipuler ce type de fichier pour les application concrète demander par mes supérieurs.

6. Tkinter Widgets :

- utilisez des widgets Tkinter tels que `Listbox` et `Text` pour créer une interface utilisateur.

- Justification : Tkinter fournit une bibliothèque simple pour créer des interfaces utilisateur graphiques. Les widgets Tkinter sont adaptés pour créer des fenêtres, boutons, listes, etc.

7. Types de données simples (int, str, list) :

- utilisez des types de données simples tels que `int` et `str` pour représenter des entiers et des chaînes de caractères dans vos calculs.

- Justification : L'utilisation de types de données simples est appropriée lorsque l'on travaille avec des valeurs individuelles et que la complexité d'autres structures de données n'est pas nécessaire.

Dans ce code nous utilisons 2 interface graphique différentes une tkinter pour coller avec le cadre du projet python puis une sur html pour avoir un rendu plus agréable mais également pour manipuler des structures de code différentes et ainsi s'exercer sur d'autre chose en lien avec de la pluri discipline.