Documentation

Documentation pour l'installation et l'utilisation du projet "scale-pixel-to-cm" développé par Florian Delconte, encadré par Phuc Ngo et Isabelle Debled-Rennesson.

Contents

1 Présentation

1.1 Conseils de lecture

Les textes écrits en *italique* sont des chemins de répertoire, ou des noms de variables.
Les lignes débutant par $ sont des commandes permettant l'installation de dépendance ou l'exécution de script.

1.2 Introduction

Le stage a débuté le 17 mars et s'est terminé le 17 septembre. Durant cette période nous avons développé un ensemble de module qui, appelé successivement, permettent de calculer l'échelle de conversion de pixel à cm pour en ensemble d'image. Cette méthode nécessite qu'un damier soit présent dans l'image. Les résultats sont écrits dans un fichier txt. Ce document présente les dépendances nécessaires au projet ainsi qu'une manière de les installer pour un ordinateur fonctionnant sous Linux. Il présente également les interactions entre les différents scripts développés et une façon de les exécuter.

1.3 Vu global du projet

Le projet est divisé en trois modules (voir 1.1):

- **Module de segmentation** (*compute_scale/segmentation_mire/*) : Ce module est codé en python. Il permet d'entraîner un modèle de segmentation de grume ou de mire. Il permet également de prédire la segmentation sur un set d'image. Un modèle entraînée à segmenter la mire est fourni avec le projet. Il est situé dans *compute_scale/segmentation_mire/modele/save/400_M_16.hdf5*.
- **Module espace de Hough** (*compute_scale/pixel_Hough/*) : Ce module est codé sous Matlab. Il permet d'obtenir les coordonnées des pixels de la mire pour un set d'image et ses segmentations. Les coordonnées des pixels sont regroupées en deux classes, les pixels appartenant aux droites horizontales de la mire et ceux appartenant aux droites verticales.
- **Module reconstruction de mire** (*compute_scale/segment_flou/*) : Ce module est codé en C++. Il permet de reconstruire à partir des coordonnées des pixels obtenue par le module espace de Hough, la mire, avec des segments flous. Ce module calcul le nombre de pixel le long des côtés des carrés de la mire comme l'intersection des droites centrales des segments flous.

Le lien entre ces trois modules se fait par l'intermédiaire de donnée temporaire (les répertoires dont le nom est en majuscule). Le répertoire */compute_scale/IMG_SGM/* contient la segmentation de la mire de chaque image. Le répertoire */compute_scale/PIXELS/* contient les coordonnées des pixels de la mire pour chaque image, ces coordonnées sont stockées sous deux fichiers, un pour les droites horizontales, un pour les droites verticales (suffixé par *_dec*). Un troisième type de données temporaire est accessible dans le répertoire *compute_scale/segment_flou/src/visu_EPS/*, il s'agit de fichier EPS permettant de visualiser la reconstruction de la mire par segment flou. Pour ouvrir ces derniers fichiers le mieux est d'utiliser Inskape ou Gimp. Enfin, le répertoire *compute_scale/SCALE/* contient les fichiers résultats sous la forme :
nom image traitée — nombre de pixel le long d'un côté du carré de la mire.

Nous allons maintenant étudier plus en détail chacun des trois modules.

Figure 1.1: Arboresence

1.4 Module de segmentation

Ce module contient un ensemble de script python permettant d'entraîner un modèle de segmentation selon l'architecture U-NET. Les images permettant de faire l'apprentissage subissent une augmentation de donnée. Il permet également de faire la prédiction sur un ensemble d'image. Ci dessous, nous expliquons le rôle des trois scripts principaux de ce module :

- "modele.py" : code de l'architecture U-Net. Tous les paramètres du réseau sont définis dans ce script. *Height*, *width* permet de gérer la taille des images en entrée. *channels* permet de définir le nombre de canal des images (3 pour les images RGB, 1 pour les images en niveau de gris). *batch_size* est le nombre d'image considéré par le réseau avant de mettre à jour les poids du modèle. Le paramètre *epochs* gère le nombre d'époque au bout duquel l'apprentissage se termine. *steps_per_epoch* est le nombre d'étape par époque. C'est aussi dans ce script que les chemins vers les bases d'entraînement/validation/test sont défini.

- "run.py" : permet de lancer l'apprentissage d'un modèle. Le modèle est sauvegardé dans le répertoire *segmentation_mire/model/save/*. On définit les paramètres de l'augmentation de donnée dans ce script. Pour lancer l'apprentissage voici la commande qu'il faut exécuter dans un environnement virtuel :

```
$ python3 run.py
```

- "predict.py" : Ce script permet de faire la segmentation sur un ensemble d'image. Il nécessite qu'un modèle soit préalablement entraîné et stocké dans le répertoire *segmentation_mire/model/save/*. Les images segmentées sont sauvegardées dans le répertoire */compute_scale/IMG_SGM/*. Voici la commande à exécuter pour faire la prédiction :

```
$ python3 predict.py <pathInputImg> <pathOutput>
```

1.5 Module espace de Hough

Ce module est constitué d'un ensemble de script Matlab permettant d'extraire les pixels de la mire et de les écrire dans */compute_scale/PIXELS/*. On génère deux fichiers pour chaque image. Le premier contient les coordonnées des pixels appartenant aux droites horizontales de la mire. Le deuxième contient les coordonnées des pixels appartenant aux droites verticales de la mire (ce dernier fichier est suffixé par *_dec*) voir 1.2.

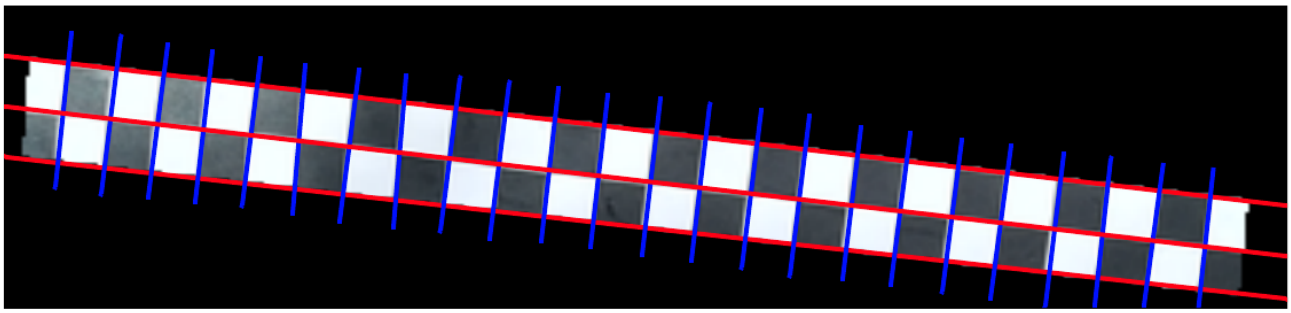

Figure 1.2: droites horizontales et verticales

1.6 Module reconstruction de la mire

Ce module est codé en C++, il utilise les librairies openCV et DGTal. Il reconstruit à partir des données dans */compute_scale/PIXELS/*, la mire avec des segments flous. Pour chaque images traité on sauvegarde dans *compute_scale/segment_flou/src/visu_EPS/* un fichier eps permettant de visualiser la mire reconstruite. Ce module écrit dans *compute_scale/SCALE/* pour chaque images sa conversion de pixel à cm. en figure 1.3 se trouve un schéma récapitulant les interactions entre les différents modules.

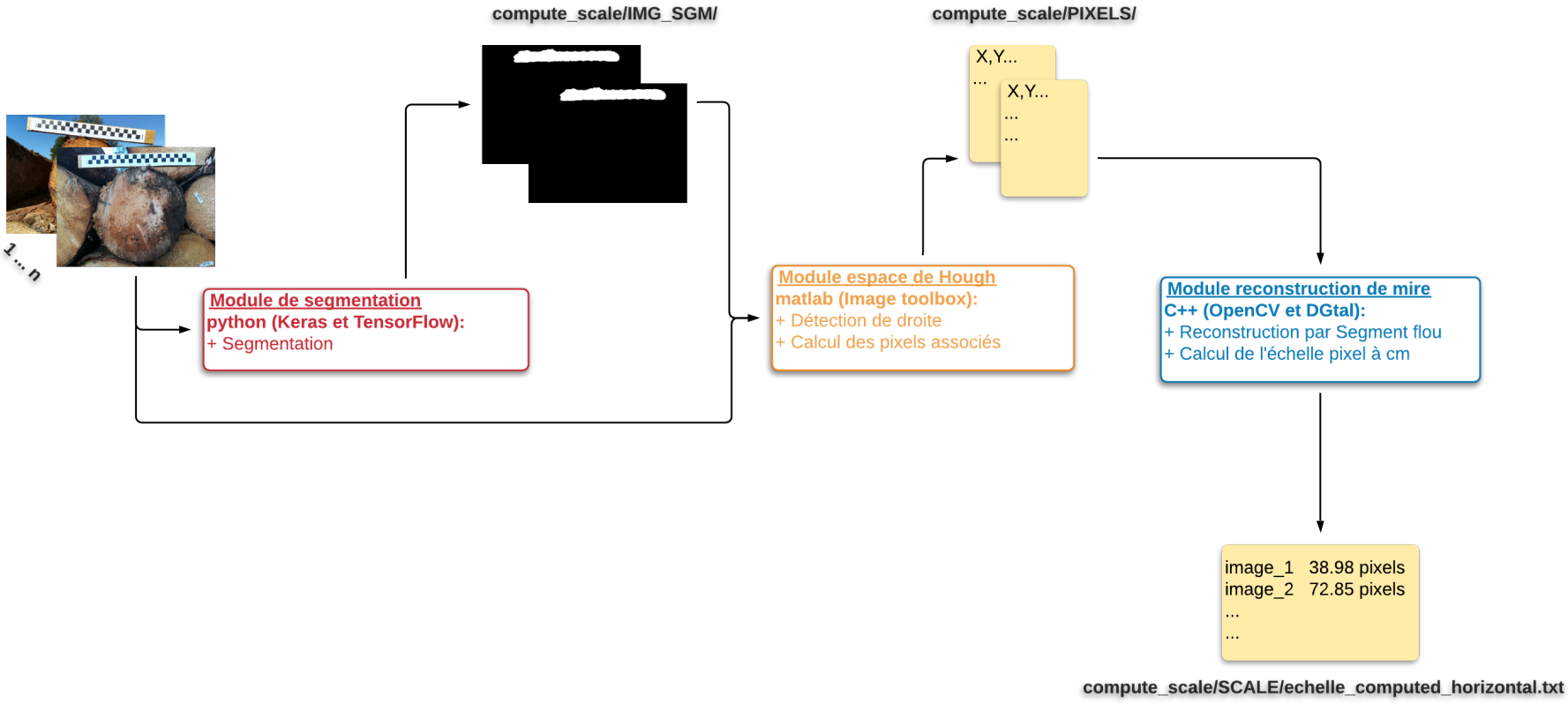

Figure 1.3: Récapitulatif des trois modules et de leurs interactions

2 Installation

L'installation des dépendances sur Debian Scretch et Ubuntu 18.04 a été testé. Ci-dessous se trouve une procédure à exécuter commande par commande pour installer l'ensemble des dépendances nécessaire au bon fonctionnement du projet.

2.1 dépendances du projet

Python3

```
$ sudo apt-get update
$ sudo apt-get upgrade python3
```

Pip3

```
$  sudo apt-get install python3-pip
```

skimage

```
$  pip3 install scikit-image
```

numpy 1.16.2

```
$  pip3 install --upgrade numpy==1.16.2
```

cv2

```
$  pip3 install opencv-python
```

Pillow

```
$  pip3 install Pillow
```

virtualenv

```
$  sudo apt install virtualenv
```

créer un environnement virtuel

```
$  cd <Path to the project install>/compute_scale/segmentation_mire/
$  virtualenv --system-site-packages -p python3 ./venv
```

Activer un environnement virtuel et installer tensorflow/keras

```
$  source ./venv/bin/activate
$  pip3 install --upgrade tensorflow
$  pip3 install keras
```

sortir de l'environnement virtuel

```
$  deactivate
```

MATLAB R2019a
L'installation Matlab est gratuite pour les enseignants et les étudiants. Voici un lien qui explique pour le personnel de l'université de lorraine les démarches à suivre pour installer Matlab :
cliquer ici

DGtal

Voici un lien qui explique comment installer DGtal, attention il y a des dépendances à installer avant de build DGtal (c++ compiler + cmake_¿3.1 + boost + zlib)
cliquer ici

Build le module SegmentFlou

Le module qui travaille avec les segments flous est codé en C++, il faut générer le makefile. Voici une suite de commande permettant de le faire :

```
$ cd <Path to the project install>/compute_scale/segment_flou/build/
$ cmake ../src
$ make
```

3 Utilisation

Pour simplifier l'utilisation de ce projet, deux script bash sont mis à disposition. L'un, "writeScale.sh" prend en entrée un répertoire d'image et le nom du fichier résultat. Il apelle les trois modules décrit précédemment. Il écrit dans *compute_scale/SCALE/NomDuFichierRésultat.txt* le nombre moyen de pixel le long des côtés de la mire pour chaque image. L'autre "clean.sh" permet de supprimer les répertoires temporaire, il faut l'utiliser après chaque utilisation de "writeScale.sh".

3.1 writeScale.sh

Ce script permet de calculer la conversion de pixel à cm pour un ensemble d'image.

```
$ ./writeScale.sh <Path to your images> <name of output file>
```

Lorsque l'exécution est terminé vous trouverez le nombre moyen de pixel le long des côtés de la mire pour chaque image dans le fichier *compute_scale/SCALE/NameOfOutputFile.txt*.

3.2 clean.sh

Ce script permet de supprimer les données temporraire généré lors de l'appelle de writeScale.sh. Il faut l'executer après chaque appel de writeScale.sh.

```
$ ./clean.sh
```

3.3 plugin ImageJ

Un plugin ImageJ nommé "segment_mesure.java" se trouve dans le répertoire *compute_scale/*. Une fois installé et exécuté, il invite l'utilisateur à sélectionner un fichier de conversion pixel à cm (généré par "writeScale.sh"). Une fenêtre de log s'ouvre. Tant que la fenêtre de log est ouverte, l'utilisateur peux ouvrir des images depuis l'interface ImageJ et y effectuer des mesures en cm directement avec les outils d'ImageJ.