|  |  |
| --- | --- |
| FRANÇOIS Tanya |  |
| SAILLY Steven | 21954116 |

# Linguistique – Rapport de projet

## Présentation et objectif

Ce projet porte sur la reconnaissance optique de caractères japonais. Le japonais présente la caractéristique d’utiliser à la fois des caractères d’origine chinoise (*kanji*), deux syllabaires dérivés des *kanji* (*hiragana* et *katakana*), et, dans une moindre mesure, l’alphabet latin (*rōmaji*). De plus, le japonais ne comprend pas d’espace, et peut s’écrire soit de haut en bas et de droite à gauche (format traditionnel, *tategaki*) ou de gauche à droite et de haut en bas (*yokogaki*). Enfin, certains kanji sont très similaires, voire impossibles à distinguer pour un œil non averti. Nous pouvons notamment citer la paire 土 - 士 (terre, *tsuchi* – guerrier, *shi*) ou le quadruplet か - カ - 力 - 刀 (*hiragana* « ka » - *katakana* « ka » - puissance, *chikara* – épée, *katana*). De ces particularités naissent des difficultés à extraire correctement les caractères d’un texte écrit en japonais.

Nous avons travaillé sur la reconnaissance de l’écriture manuscrite sous deux formes : l’une sur un support numérique, l’autre sur un support traditionnel. D’une part, nous avons réalisé une fenêtre permettant de tracer un unique caractère, et présentant les caractères les plus ressemblants. D’autre part, avant de procéder à la reconnaissance en tant que telle, nous extrayons chaque caractère avant de le soumettre au processus de reconnaissance. Le processus de reconnaissance consiste en la comparaison de l’image du caractère avec des images de référence (voir section « Bibliothèques et ressources utilisées »). Par nature, ce processus est limitant puisque le contexte n’est pas pris en compte.

Notons qu’afin de faciliter l’extraction des caractères sur support traditionnel, nous avons opté pour nous limiter à des textes écrits sur des *genkō yōshi*. Sur ces feuilles à carreaux, chaque carreau correspond à un caractère. Enfin, afin de ne pas alourdir inutilement les répertoires contenant les images de référence, nous nous sommes limités, dans la version finale, à environ 100 caractères sur lesquels la comparaison aura effectivement lieu.

## Utilisation

Ce projet a été réalisé sous Java 11. Le répertoire resources/ ainsi que le fichier ocr\_japonais.jar sont nécessaires à l’utilisation.

Depuis l’invite de commande et depuis la racine du projet :

- pour la version « support numérique » : java - jar ocr\_japonais.jar

- pour la version « support manuscrit » : java - jar ocr\_japonais.jar [fichier\_à\_traiter]

Le fichier à traiter doit être placé dans le répertoire resources/handwriting/. Pour utiliser le fichier d’exemple resources/handwriting/src.png, il suffit d’entrer un nom de fichier inexistant.

## Structure

Outre la classe Main permettant de lancer la version demandée par l’utilisateur, le projet se sépare en cinq classes, dont une partagée.

* La classe App permet de lancer l’interface graphique et de la mettre à jour en fonction de ce qu’écrit l’utilisateur. L’interface graphique se sépare en deux parties : à gauche se trouve le panneau où l’utilisateur peut écrire, tandis qu’à droite sont donnés les caractères les plus ressemblants au caractère en cours d’écriture, ainsi que leur taux de ressemblance avec ce caractère. L’attribut ArrayList<Kanji> kanji contient les caractères que représentent les images de référence.
* La classe Kanji contient notamment les attributs String kanji et double similarity, qui représente le taux de similarité d’un caractère avec le caractère en cours d’écriture par l’utilisateur. L’attribut kanji est initialisé grâce au nom du fichier de l’image de référence que l’objet va représenter : pour tout fichier *f* représentant un caractère *c*, le nom de *f* est le codage Unicode de *c*.
* La classe SVGtoPNG contient toutes les méthodes permettant de rendre utilisables les images de référence.
* La classe PngManip contient les méthodes qui permettent de passer de l’écriture sur l’interface graphique à un fichier PNG, et de calculer la similarité entre deux images. **TODO expliquer le calcul**
* La classe Handwriting permet d’extraire, via la transformée de Hough, les caractères présents dans l’image.

## Bibliothèques et ressources utilisées

Les images de références sont issues du projet KanjiVG.

Afin de réaliser au mieux les différentes fonctionnalités, trois bibliothèques ont été utilisées.

* Apache Batik
* JavaFX
* OpenCV

## Résultats et limites

Complexité de la reconnaissance support trad pas optimisée => peut prendre bcp tps

## Extensions envisageables