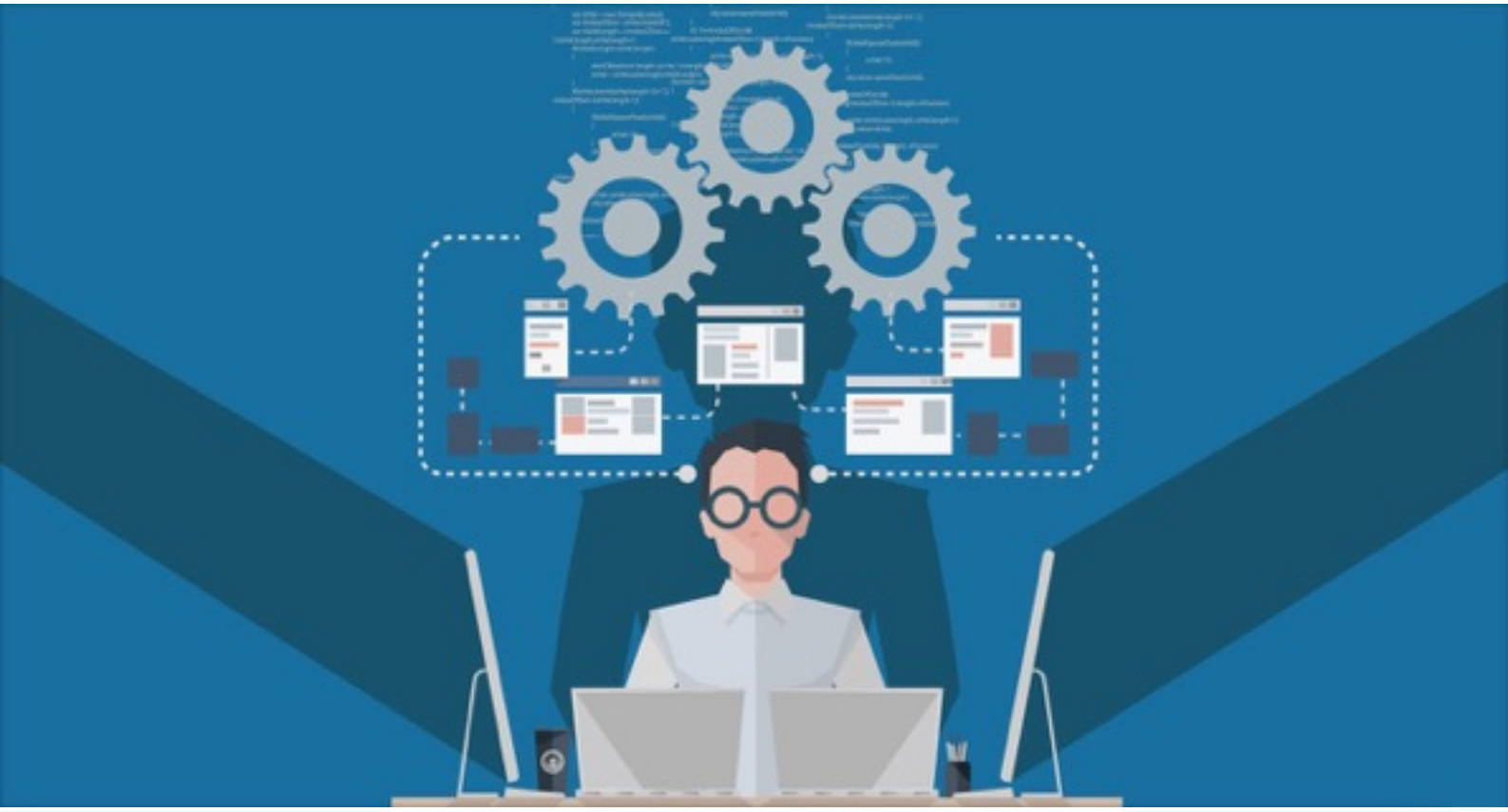
IMT Lille Douai
École Mines-Télécom
IMT-Université de Lille

Image Vision®

Rapport - PIIM



Reconnaissance de logo via une application Android

FOVET Louis
CLOUET Benjamin

Le projet

Au cours du module « Projet Informatique Image Mobile », nous avons été amenés à développer une application Android capable d'identifier le logo d'une marque dans une photo capturée, ou sélectionnée dans la bibliothèque d'un smartphone. Une fois la marque identifiée, l'application peut servir à diriger l'utilisateur vers le site web de la marque.

Le développement de l'application s'est déroulé en trois parties:

- La première partie consistait en une version simplifiée de l'application. La reconnaissance de logo se fait alors via détection et comparaison de leurs caractéristiques par rapport à une base d'images de référence, stockée localement dans le téléphone.

- La deuxième partie avait pour objectif d'étendre la version initiale afin d'utiliser une base d'images hébergée par un serveur distant. Cette fois ci la reconnaissance d'images est faite à l'aide d'une méthode d'apprentissage automatique appelée classification. Une fonctionnalité de rognage de la photo a également été mise en place.

Du point de vue de l'utilisateur

Concernant l'utilisateur, l'application lui permet de prendre une photo, ou d'en sélectionner une depuis la bibliothèque de son smartphone. Il peut également rogner cette photo, pour ne conserver que la partie qu'il souhaite analyser.
Il soumet ensuite à analyse l'image sélectionnée. S'il s'agit d'un logo enregistré dans notre base de données, et donc connu de l'application, alors celle-ci renverra l'image de référence de la marque correspondante. De plus, nous proposons à l'utilisateur de le rediriger, s'il le souhaite, vers le site web de la marque.

Environnement logiciel

Afin de mener à bien ce projet deux choix s'offraient à nous comme IDE à savoir Eclipse et IntelliJ. Nous avons choisi d'utiliser IntelliJ comme IDE, JavaCV comme bibliothèque et Github comme plateforme d'hébergement de notre code.
Nous avions déjà utilisé IntelliJ et Github auparavant, que ce soit en milieu professionnel ou bien pour les projets informatiques menés en première année. Ainsi, l'environnement nous était familier et nous pouvions sereinement débuter le projet sans apprentissage des technologies au préalable.
Pour JavaCV, le choix nous a été suggéré par les enseignants du module, nous avons donc suivi leur avis et intégré cette bibliothèque au projet, non sans difficultés.
Le choix de cette bibliothèque s'est a finalement été judicieux puisqu'elle s'est révélée très efficace et adaptée à nos besoins pour réaliser les fonctionnalités demandées.

Classes utilisées

MainActivity

Activité principale, sur laquelle s'ouvre l'application. C'est à cet endroit que sont déclarés les intent nécessaires à la capture/sélection de photo. Le rognage de la photo est également présent dans cette classe.
Une redirection vers AnalysisActivity est effectuée lorsque l'on effectue l'analyse.

AnalysisActivity

Activité qui affiche le résultat de notre analyse. Sur cette activité est affiché l'image de référence de la marque associée, son nom, ainsi qu'une possibilité d'être redirigé vers le site web de la marque.

Utils

Dans cette classe, on trouve des méthodes qui ont un objectif assez général, dans le sens où elles seront utilisées plusieurs fois, à des endroits différents du cycle de vie de notre application. Ainsi, nous retrouvons des méthodes de mise en cache d'image, de mise en cache de string, mais aussi la réduction de la résolution d'une image.

CallServer

Cette classe, dont un objet est appelé dès le lancement de l'application, va permettre d'effectuer toutes les interactions avec le serveur. C'est lui qui va appeler notre classe Json, afin de créer une cascade d'appels pour récupérer toutes les informations nécessaires.

Json

Un objet de la classe Json est instance dans le constructeur de CallServer. C'est elle qui va récupérer index.json, qui décrit l'ensemble des informations à récupérer. C'est aussi elle qui va parser le Json récupéré, et créer une liste de marques (classe Brand).

Brand

Pour chaque marque présente dans le fichier index.json, on va créer un objet de type Brand. Cet objet contiendra toutes les informations concernant cette marque: nom, site web, image de référence, et classifier. Une fois l'objet créé, il lance lui-même les requêtes de récupération du classifier et de l'image de référence sur le serveur.

Compute

La classe Compute permet d'effectuer toute l'analyse entre notre photo et celles stockées en base de données. Cette analyse est faite via l'algorithme de détection SIFT, puis en faisant une analyse entre les points d'intérêts détectés sur notre image et les classifiers récupérés sur le serveur.

AssetManager

La classe AssetManager n'est plus utilisée dans la version actuelle de l'application. Lorsque nous étions à la version où il fallait charger les assets en mode local, elle permettait de tout charger à l'initialisation, de générer les descripteurs, et d'effectuer l'analyse entre l'image choisie et les différentes images. Cela dans le but de trouver celle qui correspondait au mieux.

Difficultés rencontrées

Nous avons rencontré plusieurs difficultés au cours de ce projet, notamment lors de l'intégration de la bibliothèque JavaCV.
Nous avons eu des difficultés également avec la méthode load() de l'image: en effet, photoURI.getPath() ne renvoyait pas un chemin correct. Afin d'y remédier, nous avons directement transmis le chemin du fichier créé lors de la prise de la photo.

Suite à cela, nous pouvions correctement prendre des photos et les soumettre à l'application. Cependant nous ne pouvions pas sélectionner de photos depuis la bibliothèque, car le load ne fonctionnait pas correctement pour cette partie: il s'est avéré que l'URI était mal récupérée.

La dernière erreur réellement bloquante que nous ayons rencontrée est la suivante :

01-05 18:33:38.434 4553-4553/fr.fovet.logorecognition E/cv::error(): OpenCV Error: Insufficient memory (Failed to allocate 127844356 bytes) in void cv::OutOfMemoryError(size_t), file /home/saudet/projects/bytedeco/javacpp-presets/opencv/cppbuild/android-arm/opencv-2.4.11/modules/core/src/alloc.cpp, line 52*

Cette erreur se produisait une fois que l'objet SIFT avait été créé. L'erreur de mémoire était renvoyée par le sift.detect(), qui nous sert à la détection des points d'intérêt.

Après un certain laps de temps passé à être bloqué sur ce point, nous nous sommes rendu compte que le problème venait de la résolution des photos: en effet, celles-ci ayant une trop grosse résolution, la détection des points d'intérêt levait une erreur de mémoire.

Nous avons finalement résolu le problème en imposant une taille maximale d'image. Ainsi, si l'image est trop grande, elle est automatiquement rétrécie pour être exploitable sans saturater la mémoire du terminal.

Connaissances acquises

Lors de ce module nous avons appris à nous familiariser à la bibliothèque JavaCV.
Nous avons pu nous initier au travail sur la détection d'images, un domaine que nous n'avions pas exploré jusqu'alors. Nous avons pu perfectionner nos méthodes de programmation notamment en matière d'optimisation et de clarté du code.

Nous avons aussi appris à nous synchroniser et nous coordonner malgré la distance, ce qui n'a pas toujours été une tâche facile, notamment lors de la période de fin d'années avec les fêtes.

Enfin nous nous sommes rendu compte que la communication était essentielle sur un tel projet, particulièrement lorsque nous rencontrons un point bloquant. Cela évite de rester coincer sur une même erreur sans pouvoir avancer dans le projet pendant plusieurs jours.

Bilan

Au terme de ces presque 6 mois de projet, nous disposons d'une application fonctionnelle, à laquelle nous continuons d'ajouter des fonctionnalités, comme la fonction rogner par exemple.
Au départ, le projet fut assez simple à mettre en route, mais nous avons rapidement rencontré des difficultés, principalement sur la deuxième version de l'application.

Nous avons mis plus de temps que prévu initialement à terminer l'application, rendant ce projet assez complexe mais très intéressant d'un point de vue développement. En effet, nous ne travaillons pas sur ces méthodes et ces techniques habituellement, l'environnement JavaCV n'étant pas un outil auquel nous étions accoutumés. Nous avons donc pu nous familiariser à un nouvel environnement de développement qui pourra potentiellement nous être utile pour notre travail en entreprise.

Nous avons aussi particulièrement apprécié l'esprit de groupe entre tous les membres du profil informatique. Un esprit collaboratif s'est rapidement installé, et nous avons pu travailler dans la bonne humeur.