**Logo Compare - Rapport**

Traitement d’image - Projet 2ème semestre

3ème année, DLM 2015 - 2016

# Dates

**Date du document :** 19 avril 2016  
**Début du TP :** 23 février 2016  
**Remise des délivrables :** 10 mai 2016   
**Présentation du projet :** 10 mai 2016

# Informations

**Etudiants :** Di Stasio Leonardo & Assunçao Jeshon  
**Professeur :** Tièche François

# Introduction

L’objectif de ce projet est de permettre à un utilisateur d’une application de prendre en photo un logo avec l’appareil photo du smartphone et de comparer cette image avec la bibliothèque de logo déjà existante dans l’application, comportant des zones-clefs pré-calculés, afin de faire, à terme, de la reconnaissance de logo.

Pour cela, nous avons utilisé l’algorithme « SIFT » permettant de rechercher et détecter des zones d’intérêt dans une image, qui dans notre cas sera un logo.

# Mode d’emploi

# Technologie

Pour ce projet, nous avons utilisé l’algorithme SIFT qui fait partie des algorithmes du domaine de la vision par ordinateur, afin de détecter et identifier les éléments similaires entre différentes images.

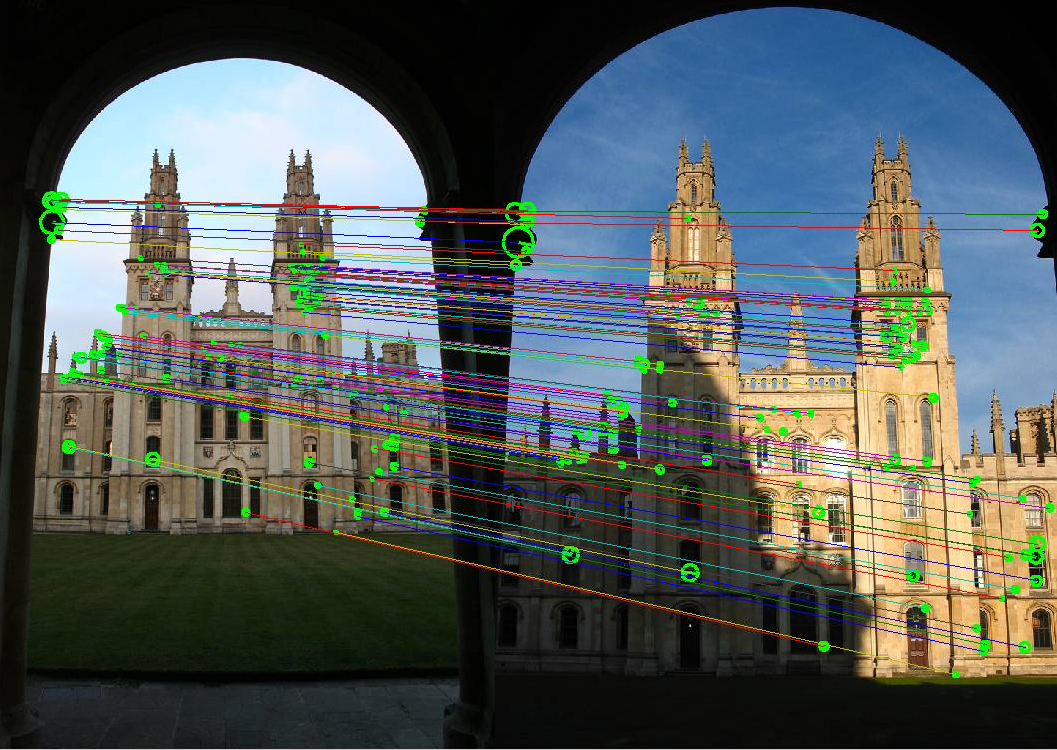
L’étape fondamentale de cet algorithme consiste à calculer ce que l’on appelle les « descripteurs SIFT » des images que l’on traite. Il s’agit d’informations caractérisant le contenu de cette image de la façon la plus indépendante possible de l’échelle, du cadrage, de la luminosité et de l’angle d’observation. Ce qui permet donc à deux images d’un même objet ou d’un même lieu, d’avoir beaucoup de chance de contenir les mêmes descripteurs SIFT et ainsi de pouvoir affirmer qu’il s’agit du même lieu ou du même objet que l’on peut observer sur les deux images.

Figure 3.1 : Exemple d'utilisation de l'algorithme SIFT

L’algorithme se divise en deux grandes étapes :

1. **Calcul des points d’intérêts et des descripteurs**: Cette étape permet de transformer une image en vecteurs de caractéristiques.
2. **Mise en correspondance**: Cette étape consiste en la comparaison des descripteurs de deux images afin de détecter un objet en particulier ou de trouver quelle transformation a subi l’image.

La première étape de l’algorithme peut se diviser elle en 5 autres étapes :

1. **Construction de l’espace des échelles**: Permet de justifier l’invariance des descripteurs par rapport à la mise à l’échelle.
2. **Localisation des extrema dans les DoG**: Permet d’approximer le laplacien grâce à des différences de gaussiennes et détecter les extrema locaux dans les contours.
3. **Localisation précises des points clés / Elimination des points à faible contraste et des points sur les arrêtes**: Permet de détecter des coordonnées exactes des points d’intérêt grâce à une interpolation.
4. **Affectation d’orientation aux points clés**: Permet de calculer des histogrammes d’orientations en fonction du voisinage. Permet de également de justifier l’invariance des descripteurs par rapport à la rotation.
5. **Calcul des descripteurs**: Permet de générer des vecteurs descripteurs associés à chaque point d’intérêt.

# Problèmes rencontrés

## Problème 1

Nous avons essayé de comparer deux images mais sans succès. Le problème de cette API est qu’elle est censée tourner sur un PC avec toutes les fonctionnalités de JAVA. Nous nous sommes rendu compte qu’il nous manquait différentes librairies pour permettre la comparaison entre deux images et donc nous avons dû utiliser une méthode nettement moins précise comme la distance Euclidienne.

Cette méthode que nous avons vue en cours permet de calculer la distance entre deux images. Elle nous donnait des résultats avec une certaine distance mais ne prenait pas en compte la disposition ou encore l’orientation des différents Features calculés grâce à la librairie.

De plus il nous a fallu fixer un pallier de distance pour dire à l’utilisateur qu’aucun résultat n’a été trouvé. Ce pallier a été placé au cours de différents tests et est donc très peux précis.

## Problème 2

Un autre problème rencontré était le fait de calculer plusieurs images d’un coup avec l’algorithme SIFT. En effet quand nous photographions un objet, on calculait tous les points des images qu’on avait stockées dans l’application. Cependant, un smartphone est nettement moins puissant qu’un ordinateur et il nous a fallu régler ce problème.

Pour éviter de surcharger l’appareil nous insérons nos propres images à l’aide d’un formulaire dans l’application, nous calculons les points grâce à l’algorithme SIFT et nous les stockons dans une base de données locale.

Une fois que l’utilisateur prendra en photo un logo, nous aurons déjà une BDD pré faite qui permettra d’aller plus vite et de ne pas faire cracher l’appareil à cause de la mémoire.

# Références

* DEVELOPPEZ.COM. Utiliser et appliquer l’algorithme SIFT sur une image avec Android [en ligne]. Disponible sur : <http://jodul.developpez.com/tutoriels/android/utiliser-sift/>. (19.04.16)
* PHILIPPINE POUBLANG. L’algorithme des SIFT [en ligne]. Disponible sur : <https://sites.google.com/site/poublangsift/l-algorithme-des-sift>. (19.04.16)
* WIKIPEDIA. Scale-invariant feature transform [en ligne]. Disponible sur : <https://fr.wikipedia.org/wiki/Scale-invariant_feature_transform>. (19.04.16)