Traitement des Images et Reconnaissance des Formes

Denis Castéran - caster_d
Hugo Rybinski - rybins_h

Juillet 2017
Epita - SCIA

Table des matières

Introduction

Nous avons décidé pour se projet de réaliser un programme qui, à partir de photos de référence, détecte dans d'autres photos la présence de canette d'une marque de jus de fruit. Bien sûr, ces photos peuvent contenir d'autres objets, et avoir des conditions d'éclairages variées.

Pour faire cela, nous avons dans un premier temps calculer les descripteurs SIFT pour l'image de référence et pour l'image testée, puis nous avons utilisé diverses techniques pour mettre en relation les descripteurs qui correspondaient.

Partie 1

Calcul des descripteurs SIFT

La scale-invariant feature transform (SIFT) est un algorithme permet de d'identifier les éléments similaires entre différentes images. Il a été développé en 1999 par D. Lowe.

Le but de l'algorithme est de calculer les "descripteurs SIFT" des images à comparer. Ces descripteurs sont calculer à partir de sous-images locales, de manière à ce que l'échelle, la luminosité, le cadrage et la qualité de la photo ne les modifient pas.

Deux images similaires partageront un grand nombre de descripteur, alors que deux images différentes en auront très peu.

1.1 Calcul des octaves

La première étape est de calculer une pyramide gaussienne. Cette pyramide est composée de 4 octaves et de 5 niveaux d'échelles.

Chaque octave contient 5 niveaux d'échelle. Chacune de ces octaves débutent

à partir d'une image, la n^{ieme} octave débutant avec l'image de départ réduite $n-1$ fois. Le facteur de réduction que nous avons décider de prendre est$\sqrt{2}$. Ainsi une octave n aura comme première image l'image de départ réduite à l'échelle$\sqrt{2}^{n-1}$. Le but de cette réduction est de simuler la modification du facteur d'échelle, permettant ainsi de reconnaître les points caractéristiques des éléments dans l'image quelque soit leur taille.

Le premier niveau d'une octave est la première image de départ déterminée à partir de l'image de départ comme dit précédemment. Chaque niveau qui suit correspond au niveau précédent auquel on a appliqué un flou gaussien.

On se retrouve donc avec 4 octaves, chacune commençant avec une image d'une taille différente, et 5 niveaux d'échelles dans chaque octave, chacun de ces niveaux étant plus flou que le précédent.

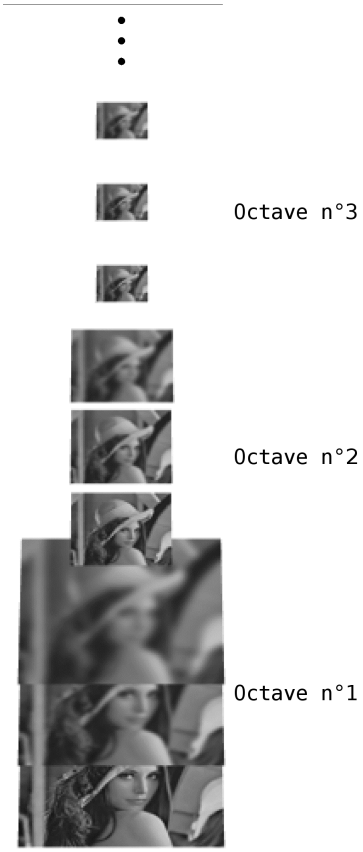

FIGURE 1.1 – Exemple de pyramide gaussienne avec 3 octaves et 3 niveaux d'échelles

Le montant de 4 octaves et 5 niveaux d'échelles peut en théorie varier, mais nous avons décider de partir sur ces chiffres car ils sont ceux conseillés par le créateur de l'algorithme SIFT.

1.2 Calcul des différentielles

La seconde étape revient à appliquer des différences de gaussiennes (DoG).

Dans chaque octave, 5 niveaux d'échelles ou 5 gaussiennes sont présentes. On fait la différence entre chaque gaussienne à partir de la première gaussienne jusqu'à la dernière dans chaque octave. On se retrouve ainsi à 4 différences de gaussiennes dans chaque octave.

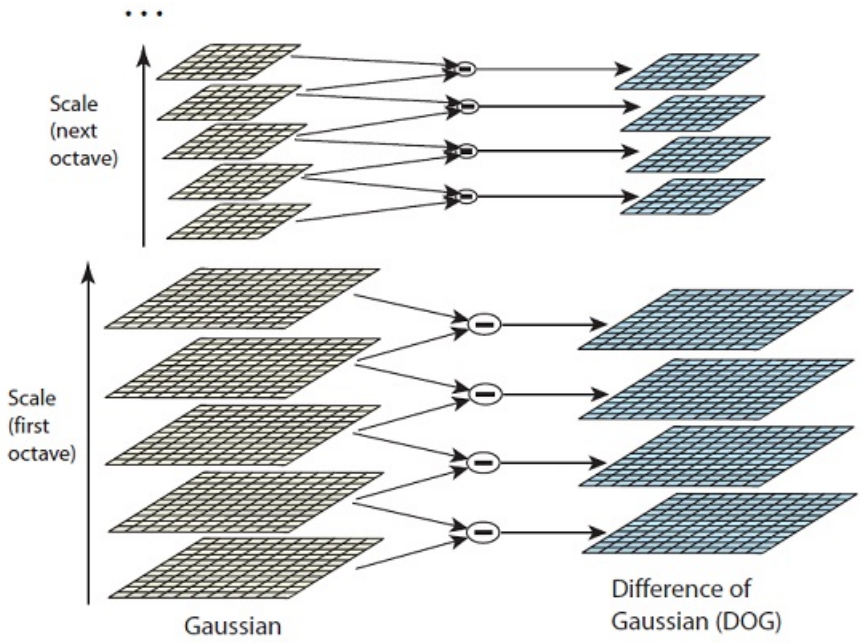

FIGURE 1.2 – Différences de gaussiennes sur deux octaves

La différence entre une image et sa gaussienne permet de mettre en évidence les bords de chaque éléments présent sur l'image. Ces différences de gaussiennes reviennent à faire une approximation d'un Laplace de gaussienne. Le Laplace de Gaussienne était assez coûteux en calcul, faire cette différence à la place permet d'accélérer fortement cette étape.

1.3 Recherche des maxima et minima locaux

Nous avons donc à ce stade 5 octaves contenant chacune 4 différences de gaussiennes (DoG). L'étape suivante a pour but de trouver des maxima et minima locaux en prenant 3 DoG consécutives. Sur chaque octave on peut donc chercher les maxima et minima sur les DoG 1, 2 et 3, ainsi que sur les DoG 2, 3 et 4.

Chaque pixel de l'image centrale est parcouru, et on cherche à trouver tous les pixels correspondant à des maxima ou minima parmi les 26 pixels l'entourant (les 8 l'entourant sur l'image qu'on parcours et les 9 pixels de l'image supérieure et inférieure situés au niveau du pixel parcouru).

On se retrouve ainsi avec une liste de points qui sont des maximas et minimas par octave.

1.4 Élimination des points clés non-pertinents

A partir de cette liste de points par octave et des images de départ de chaque octave, on applique l'algorithme d'Harris.

L'algorithme d'Harris permet de trouver les différents coins de l'image, ne sont gardés ensuite que les points de la liste précédente qui sont sur des coins. Le but ici est de retirer tous les points qui sont seulement présents sur les bords d'éléments, ces points là étant moins pertinent en tant que point d'intérêt.

Cette étape permet aussi de réduire la quantité de point d'intérêt trouvé, allégeant donc en même temps la suite de l'algorithme en terme de calcul.

1.5 Calcul des descripteurs

A la fin de l'étape précédente, on a une liste de coordonnées sur l'image des points clés pour les deux DoG du milieu de chaque octave restant. Pour calculer le descripteur de chaque point clé, on va sous diviser l'espace autour du point clé pour créer 16 fenêtres de 8 par 8.

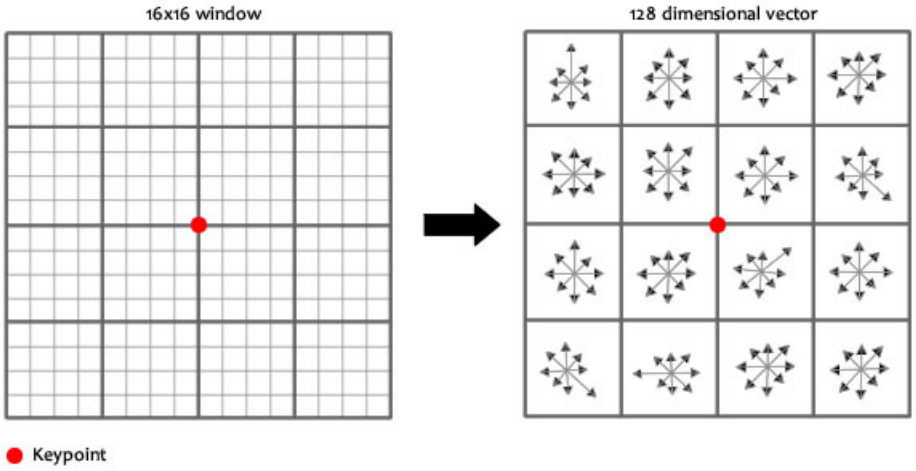

FIGURE 1.3 – Les fenêtres (de 4 par 4 ici) autour du point clé

Dans chaque fenêtre, on calcule les gradients pour chaque pixel. Selon leur orientation, on les met dans un histogramme, en arrondissant aux 20 degrés inférieurs l'orientation. On ajoute la norme dans la colonne correspondante.

Enfin, on fusionne ensemble les 16 histogrammes de 18 colonnes. On obtient un vecteur de 288 valeurs qui, associé aux coordonnées, donne un descripteur.

1.6 Persistance des descripteurs

Le calcul des descripteurs décris précédemment peut être relativement long (plusieurs minutes pour les images d'une résolution de 1920 par 860). De manière à pouvoir tester plus rapidement, nous avons décidé de sérialiser les descripteurs de l'image après leur calcul et de les sauvegarder avec les images.

Quand on lance la comparaison de deux images, soit le fichier des descripteur existe pour une des images et il est chargé, soit il n'existe pas. Dans ce cas, les descripteurs sont calculés et sauvegardés.

Ainsi, notamment quand on teste la comparaison plusieurs fois en changeant juste certains paramètres, le calcul est beaucoup plus rapide.

Partie 2

Association des descripteurs avec l'image de référence

En traitant l'image de référence et l'image testée comme décrit précédemment, on obtient une liste de descripteur pour chaque image avec leur position sur leur image d'origine et l'histogramme des gradients locaux.

L'étape suivante consiste à associer (si c'est possible) les descripteurs de la référence aux descripteurs de l'image testée pour détecter si l'image contient une canette, et si oui, où dans l'image et à quelle taille.

2.1 KDTree

Afin de calculer cette association, nous avons utilisé un KDTree. Un KDTree permet de représenter l'espace de telle manière à ce que, quand on le crée avec un ensemble de point donné, la recherche du point connu le plus proche soit beaucoup plus rapide que si on comparait l'ensemble des distance entre le nouveau point et les points connus, un à un.

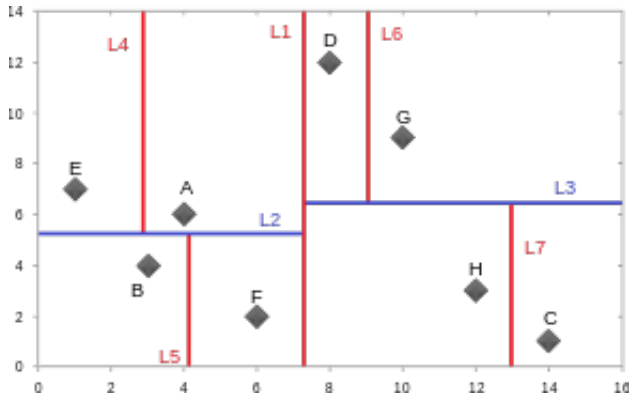

FIGURE 2.1 – Un exemple ($K = 2$) des sous-espaces construits

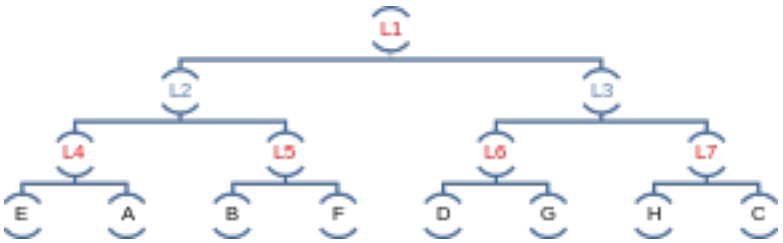

FIGURE 2.2 – L'arbre correspondant au plan

Pour cela, on va partitionner l'espace en deux de manière récursive pour avoir un point dans chaque demi espace. Ainsi, grâce à ces droites, on peut savoir à quel demi espace le point appartient et descendre dans l'arbre du bon coté de manière à ne regarder finalement la distance que des points connus du dernier sous espace.

Comme cordonnées des points, on va choisir les valeurs des histogrammes du descripteurs (donc $K = 288$ dans notre cas).

Pour chaque descripteur de l'image testé, on va garder que le descripteur le plus proche de la référence. Si la distance entre les deux descripteurs correspondant est supérieur à un certain seuil, la correspondance ne sera pas conservée.

A la fin de cette étape, on a une liste de couple de descripteur image tes-

tée/référence qui correspondent. Les descripteurs n'ayant pas trouvé de correspondance dans l'autre image sont ignorés pour la suite.

Si cette liste de correspondance est trop petite, c'est qu'il n'y a pas de canette sur l'image.

2.2 Élimination des descripteurs isolés

Nous avons constaté cependant qu'il y avait souvent quelques descripteurs isolés qui correspondaient par hasard sur l'image ailleurs que sur la canette, produisant du bruit qui rendait complexe les étapes suivantes.

Pour éliminer les descripteurs nous sommes partis de l'observation que s'il y avait une canette, alors il y avait beaucoup de descripteur sur la canette et que par conséquent, la position moyenne des descripteurs serait sur la canette.

Nous avons donc mis en place un algorithme d'expectation-maxisation pour enlever les descripteurs isolés. On calcule la moyenne des positions des descripteurs, puis on enlève les 5% de descripteurs les plus éloignés, on recalcule le centre des descripteurs restant. Tant que la distance entre les deux centres est supérieure à un seuil, on continue à enlever petit à petit les descripteurs les plus éloignés.

2.3 Calcul de la bounding box autour de la canette

Une fois la liste des descripteurs correspondant nettoyées, on va construire la bounding box autour de la canette.

On commence par construit une première boîte sur l'image de référence en prenant le x et le y minimum comme coin supérieur gauche et le x et le y maximum comme coin inférieur droite. Cela nous donne les proportions de la boîte.

Pour calculer l'échelle à appliquer pour correspondre à l'image testée, on va prendre des couples de descripteurs correspondant par deux aléatoirement (les couples (A, A') et (B, B') par exemple), regarder le ratio de la distance entre les deux points de chaque image (soit $distance(A, B)/distance(A', B')$ par exemple). En répétant un grand nombre de fois et en prenant la médiane des ratios, on obtient le coefficient d'échelle à obtenir.

Après une mise à l'échelle il faut juste placer la boîte sur l'image testée pour la centrée sur le centre calculer à l'étape précédente.

Partie 3

Résultats et critiques

3.1 Résultats

Voici un exemple de résultat produit par notre projet.

FIGURE 3.1 – L'image de test et l'image de référence utilisée

FIGURE 3.2 – Un octave

FIGURE 3.3 – Les différence de Gaussienne d'un octave

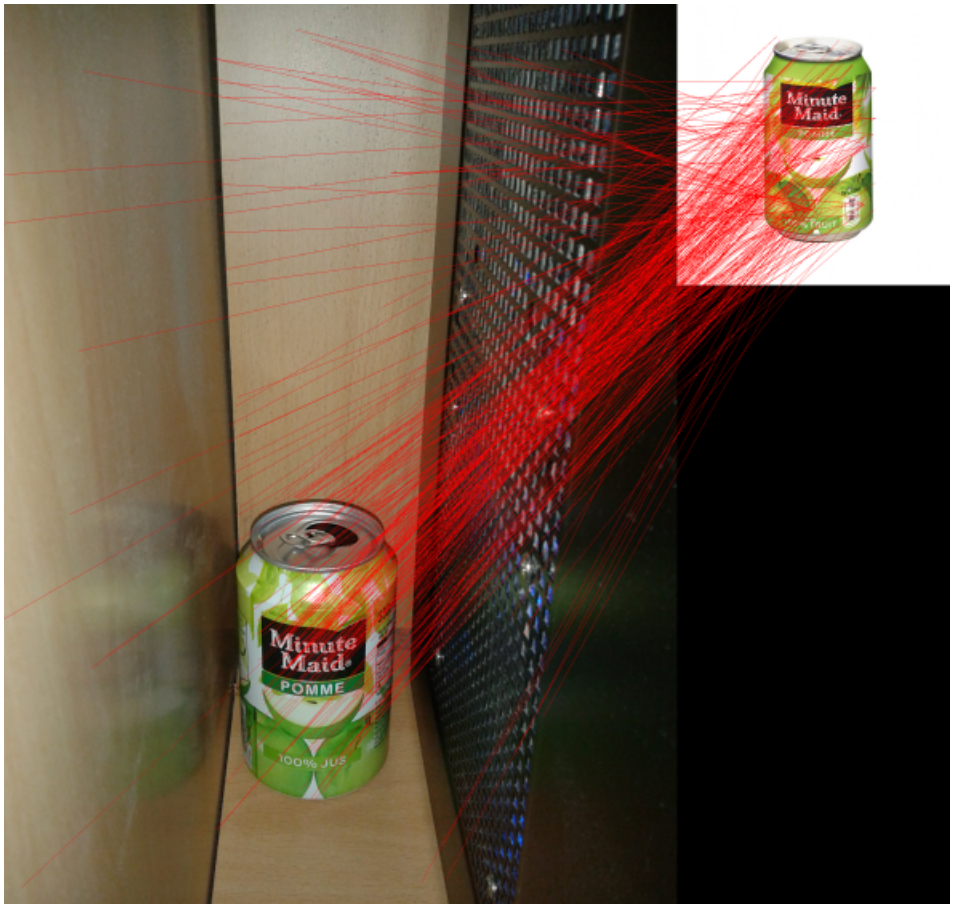

FIGURE 3.4 – Les correspondances des descripteurs avant réduction

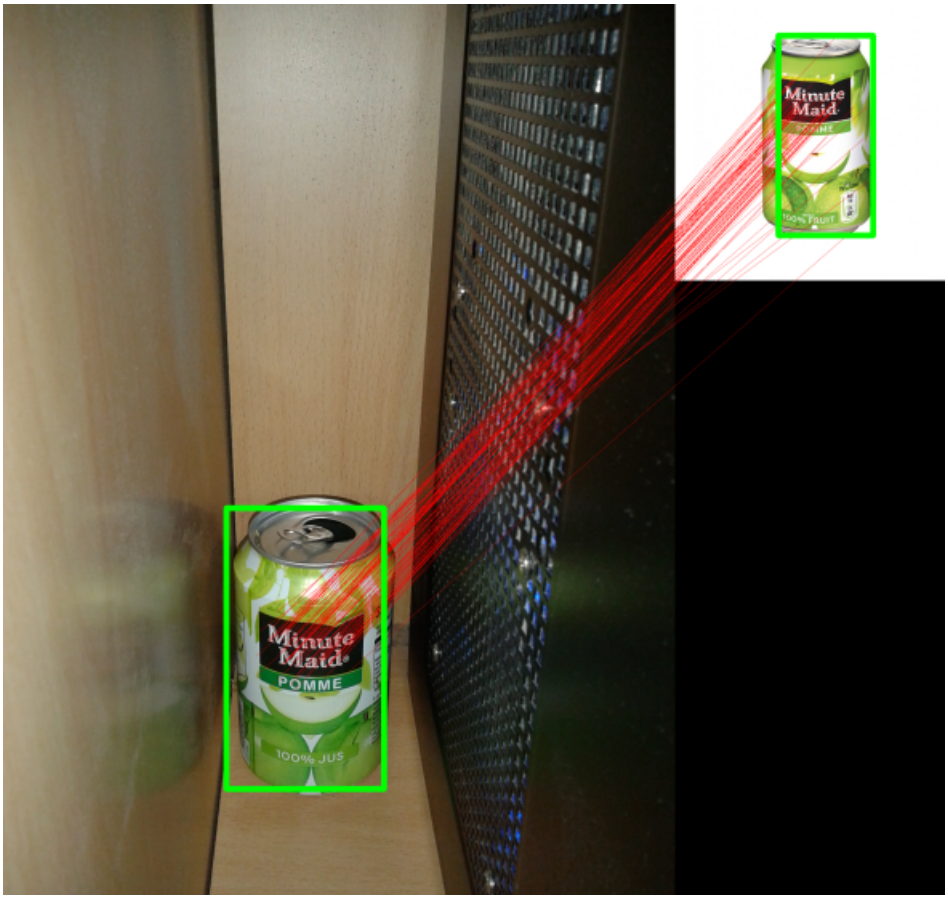

FIGURE 3.5 – Les correspondances des descripteurs finale