



# Rapport écrit TP #1

**Vision artificielle et traitement des images**

**8INF804 GR 01**

**Travail présenté**

À

**Julien Maitre**

Par

**Koffi Deladem Moïse ETOU**

**Année-2024**

**Date de remise : 22 octobre 2024**

## **PLAN**

I. Introduction

II. Les différentes Phases

1- Phase de Prétraitement

2- Phase de segmentation

3- Phase d'extraction des caractéristiques

4- Phase de mise en évidence les objets détectés

III - Problématiques / Difficultées

IV- Quelques résultats

# Introduction

L'objectif de ce TP est de concevoir un algorithme capable de détecter les changements, entre une pièce rangée (image de référence) et la même pièce mais avec du désordre (nounours à terre par exemple), et de les mettre en évidence. Nous disposons d'images de trois différentes pièces (chambre, cuisine et salon), pour chacune de ces pièces, en plus de l'image de référence, nous avons entre 4 et 9 autres images de ces pièces. Le but étant de détecter les changements sur ces images. Pour cela nous devions concevoir un seul et unique algorithme pour les trois pièces. Pour concevoir cet algorithme nous avons suivi plusieurs étapes :

1. Phase de Prétraitement
2. phase de Segmentation
3. Phase d'extraction des caractéristiques
4. Phase de mise en évidence les objets détectés

Les 4 parties ci-dessus seront décrites en détail dans ce rapport de même que les difficultés que nous avons rencontré par la suite.

## 1- Phase de Prétraitement

Cette étape est primordiale afin d'avoir la meilleure qualité possible d'image et ainsi rendre la détection des différences plus facile. Dans un premier temps nous avons choisi de redimensionner nos images afin d'avoir des images plus petites et ainsi plus rapides à traiter puis de convertir notre image en RGB puisqu'il est chargé avec OpenCV2 qui utilise le (BGR) mais que nous allons prévisualiser nos images avec matplotlib qui lui utilise le (RGB)

Après dans un second temps, nous avons créé une fonction qui permet de réaligner les images par rapport à l'image de référence. En effet, toutes les images n'ont pas été prises avec exactement le même angle ce qui peut entraîner de léger changement dans l'image. Pour cela nous avons procédé en plusieurs étapes :

1. Conversion de l'image en niveaux de gris

Nous avons converti l'image de référence et l'image à aligner en niveau de gris.

2. Détection de points clés

Grâce à l'algorithme ORB (Oriented Fast Rotated BRIEF), nous avons détecté des points clés sur l'image et leurs descripteurs. Les descripteurs sont des vecteurs qui contiennent les caractéristiques des points clés.

3. Correspondance des descripteurs.

Nous avons ensuite utilisé un match brute afin de trouver les correspondances entre les descripteurs des deux images. Les correspondances ont été ensuite triées par distance (plus la distance est petite mieux c'est).

#### 4. Sélection des bonnes correspondances

Ensuite on vient conserver un pourcentage (ici 15%) des meilleures correspondances. Toutefois, si le nombre de bonnes correspondances est inférieur à 4 on vient lever une erreur (c'est le minimum pour faire une homographie).

#### 5. Extraction des points correspondant

On vient extraire les coordonnées des points clé correspondant.

#### 6. Calcul de l'homographie

On vient calculer l'homographie (matrice de transformation), à l'aide des points correspondants à l'aide de la méthode RANSAC (RANdom SAmple Consensus) qui est robuste face aux valeurs aberrantes et au bruit.

#### 7. Application de l'homographie

On vient ensuite appliquer l'homographie. Ainsi maintenant l'image traitée correspond à l'image de référence.

Ensuite, nous avons appliqué séparément une égalisation des couleurs grâce au filtre CLAHE (Contrast Limited Adaptive Histogram Equalization) dans deux espaces colorimétriques : HSV (Hue, Saturation, Value), en nous concentrant sur la composante de valeur (V), et RGB (Red, Green, Blue). L'espace HSV a été utilisé pour mieux gérer la lumière (luminance), tandis que l'espace RGB a été privilégié pour une gestion optimale des couleurs.

Voici comment nous procéder :

Pour le CLAHE en HSV

##### 1. Conversion en HSV

Dans un premier temps nous avons converti notre image de l'espace RGB vers l'espace HSV.

##### 2. Séparation des canaux

Les trois canaux (H, S, V) sont extraits

##### 3. Application du CLAHE

L'égalisation est appliquée uniquement au canal de valeur (V) afin d'améliorer le contraste.

#### 4. Fusion et reconversion

Les trois canaux sont recombinés puis l'image est convertie en RGB.

Pour le CLAHE en RGB

##### 1. Séparation de l'image RGB en Canaux

Les trois canaux sont (R,G,B)

##### 2. Application du CLAHE

L'égalisation est appliquée sur les trois canaux.

##### 3. Fusion et reconversion

Les trois canaux sont recombinés à nouveau

Une fois le filtre CLAHE appliqué, nous avons combiné les deux images avec des poids de 0,5. Cette étape de traitement a été réalisée sur notre image de référence ainsi que sur les images de chaque chambre, ce qui est nécessaire pour détecter les nouveaux objets dans les images fournies. Le choix des espaces colorimétriques a été fait après avoir tester plusieurs autre espace comme le LAB et le YCbCr

## 2- Phase de segmentation

### - Détection de différence

Pour détecter les différences entre une image de référence et une autre, on doit commencer par se débarrasser des parties qui sont "égales" dans une image. Pour ce faire, nous avons utilisé la fonction "absdiff" qui fait la "soustraction" de deux images. Cette soustraction ne garde que les parties différentes entre les deux images. Pour effectuer la différence entre les deux images, nous avons procédé en plusieurs étapes :

##### 1. Conversion des images en niveaux de gris

##### 2. Calcul de la différence

On vient calculer la différence absolue entre les deux images en niveaux de gris

### 3. Seuil et masque

On vient appliquer un seuil afin de créer un masque binaire où seules les différences significatives sont mises en évidence (valeurs supérieures à 30). Le type de seuil appliqué est le threshold avec “*THRESH\_TRIANGLE*”

### 4. Morphologie

Des opérations morphologiques sont appliquées pour réduire le bruit dans le masque, en comblant les trous et en supprimant les petites régions isolées.

## 3- Phase d'extraction des caractéristiques

### - Detection les contours de l'image

Après les plusieurs étapes du prétraitement de l'image afin d'avoir une meilleure détection d'objet, nous avons procédé à la détection des contours dans l'image. Pour cela, nous avons simplement utilisé la fonction “*findContours*” avec le paramètre “*CHAIN\_APPROX\_SIMPLE*”.

## 4- Phase de mise en évidence les objets détectés

Pour terminer, nous avons créé une fonction de traçage de rectangle selon les contours trouvés dans l'étape précédente. Cette fonction met en évidence les objets détectés par l'algorithme.

La fonction prend en paramètre la liste de contours de l'étape précédente. Ensuite, on boucle pour chacun des contours qui ont une aire plus grande que 5500 et plus petite que 770000. Ainsi, on se débarrasse des petites et des très grosses détections parasites parmi les contours détectés.

À chaque itération de la boucle ci-dessus, nous créons un rectangle Vert à l'aide des coordonnées x et y ainsi que la largeur et hauteur trouvé avec la fonction “*boundingRect*”.

**Pipeline:** Le pipeline de notre projet est conçu de telle sorte que nous parcourons automatiquement chaque sous-dossier de chambre et effectuons sur nos images toutes les opérations mentionnées ci-dessus.

### III - Problématiques / Difficultées

Parmi les images qui nous sont fournies, certaines d'entre elles avaient des différences rendant la détection de changement plus difficile. Par exemple, l'angle dans lequel une des photos a été prise était très légèrement différent par rapport à la photo de référence. Il y avait aussi des variations d'éclairage entre d'autres photos. Par conséquent, nous devions concevoir un algorithme flexible qui prend en compte toutes ces variations.

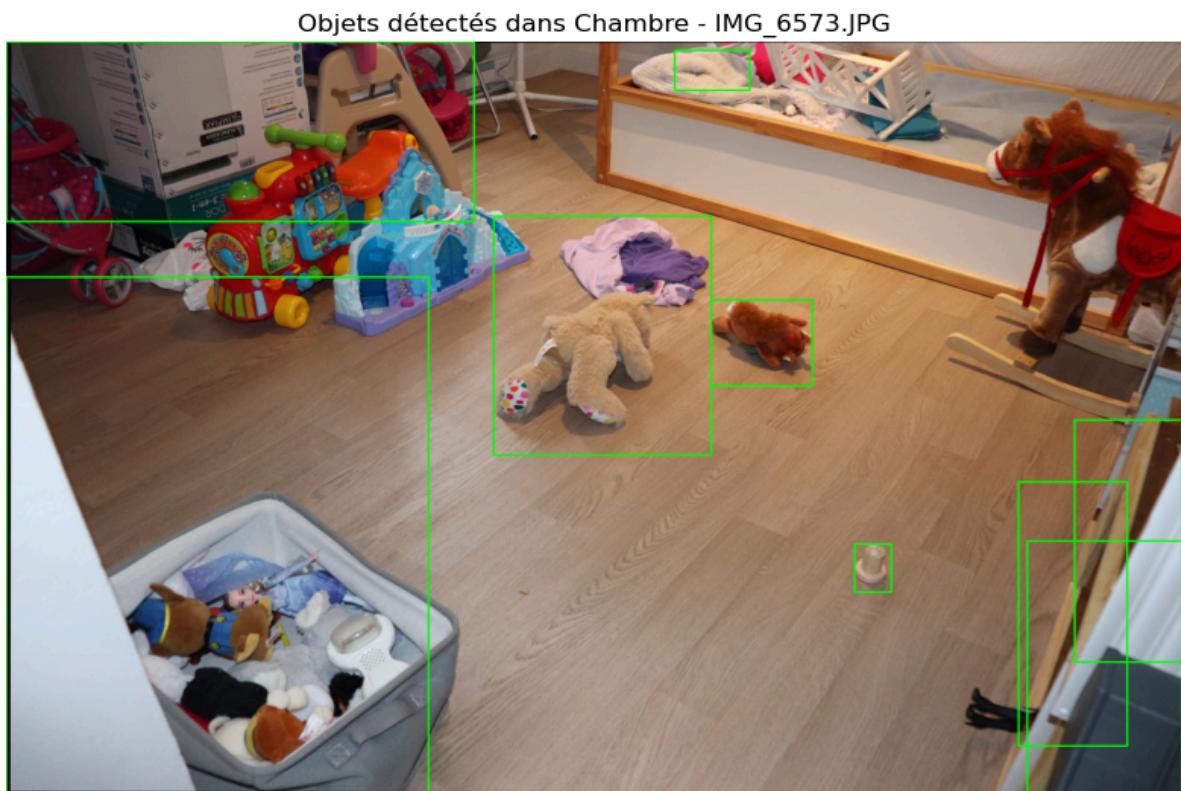
De plus, certains des objets - notamment l'ourson en peluche - avaient une couleur très similaire à l'arrière-plan. De ce fait, on devait rendre notre algorithme plus sensible aux changements ce qui a eu comme conséquence d'avoir plus de fausse détection. Pour remédier à ce problème, nous avons fait plusieurs étapes de prétraitement d'image afin de se débarrasser du bruit.

Par ailleurs, une autre difficulté résidait dans le choix des traitements appropriés à chaque image, en fonction de ses caractéristiques (luminosité, couleurs des objets, etc.). Certains traitements étaient plus adaptés à certaines images qu'à d'autres, et certaines techniques détectent donc mieux les objets sur certaines images. Il a fallu opter pour des méthodes de traitement qui étaient globalement correctes, bien que certaines aient été plus ou moins efficaces selon les cas. Nous avons dû trouver un compromis intermédiaire dans le choix des techniques. Aussi, à chaque étape du traitement, nous avons dû ajuster les paramètres par tâtonnement, ce qui rendait la tâche particulièrement fastidieuse.

## IV - QUELQUES RÉSULTATS



**figure 1- Exemple d'image de différente phase de traitement de notre image dans la chambre**



**figure 2- Exemple de résultat final de notre algorithme dans détection d'objet la chambre**



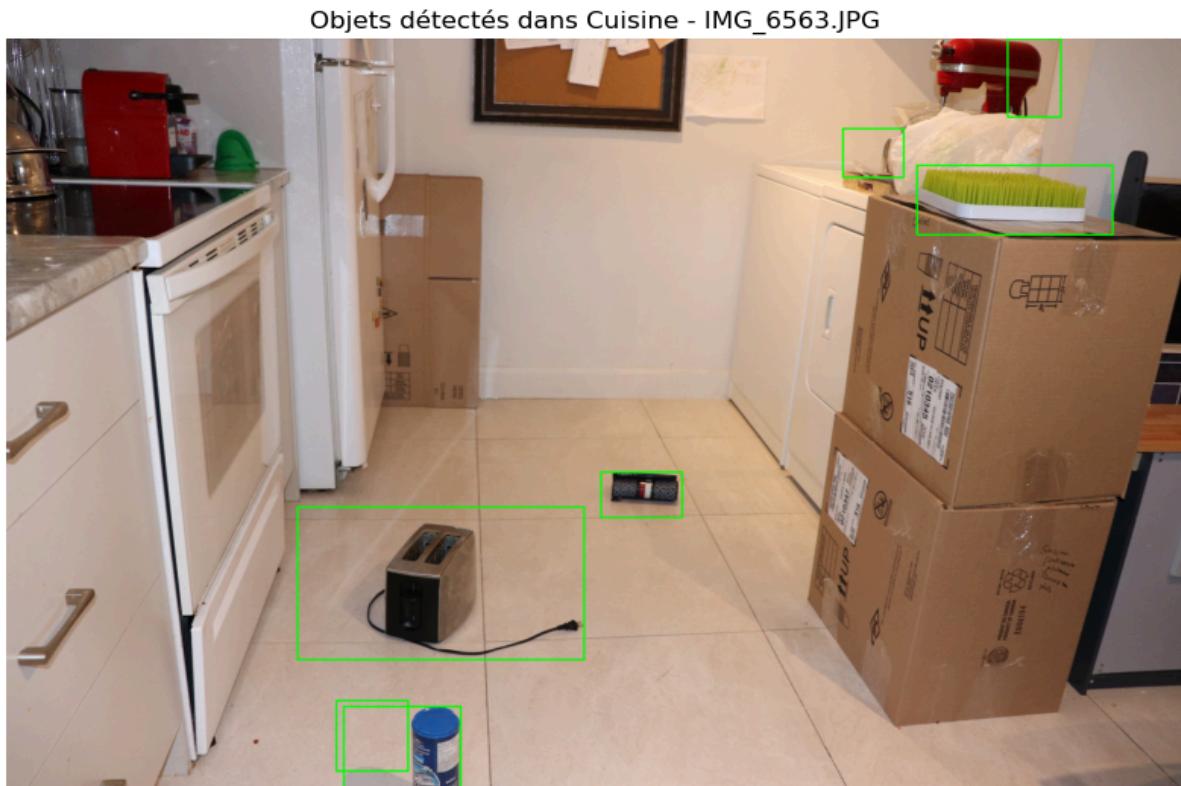
**figure 3- Exemple d'image de différente phase de traitement de notre image dans le salon**



**figure 4- Exemple de résultat final de notre algorithme dans détection d'objet dans le salon**



**figure 5- Image de différente phase de traitement de notre image dans la cuisine**



**Figure 4- Exemple de résultat final de notre algorithme dans détection d'objet dans la Cuisine**